INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Deteção de convulsões epiléticas em eletroencefalogramas usando deep learning

JOSÉ MANUEL SOARES CAMPOS FERREIRA DA SILVA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Deteção de convulsões epiléticas em eletroencefalogramas usando *deep learning*

José Manuel Soares Campos Ferreira da Silva

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: Elsa Ferreira Gomes

Júri:

Presidente:

[Nome do Presidente, Categoria, Escola]

Vogais:

[Nome do Vogal1, Categoria, Escola]

[Nome do Vogal2, Categoria, Escola] (até 4 vogais)

Porto, Outubro 2017

Família e Amigos

Resumo

Através da análise dos dados obtidos por eletroencefalogramas intracranianos (iEEG), pretende-se a identificação dos períodos com maior probabilidade de ocorrência de convulsões, permitindo dessa forma uma deteção e previsão mais eficaz dos sistemas baseados em iEEG. Sendo, com a análise dos iEEG, possível a deteção destes períodos, os pacientes apenas terão de se medicar em situações de convulsões iminentes, resultando numa redução geral dos efeitos secundários associados a este tipo de medicação.

O objetivo deste trabalho é desenvolver uma metodologia usando algoritmos de *deep learning* capaz de classificar sinais resultantes do iEEG, para deteção e previsão de convulsões epiléticas em humanos.

Pretende-se ainda que seja elaborado um estudo sobre as diferentes técnicas descritas por outros autores, de forma a avaliar a possibilidade de aplicação de algumas dessas metodologias. Pretende-se ainda comparar resultados obtidos por trabalhos já publicados com os resultados obtidos neste trabalho.

Palavras-chave: *Deep Learning, Convolutional Neural Networks*, Eletroencefalograma intracranial, Epilepsia, Previsão de convulsões

Abstract

Through the analysis of the data obtained by intracranial electroencephalograms, we intend to identify the periods of high probability of occurrence of seizures, achieving a more effective detection and prediction by iEEG based systems. If these periods can be detected with iEEG analysis, patients only have to be medicated in imminent seizures, resulting in a general reduction of the side effects associated with this type of medication.

The goal of this work is to develop a methodology using deep learning algorithms capable of classifying signals resulting from iEEG for the detection and prediction of epileptic seizures in humans.

We also intend to study different techniques used by other authors, in order to analyze and understand if it is possible to apply some of the methodologies already used and compare our results with the ones obtained by works already published.

Keywords: Deep Learning, Convolutional Neural Networks, Intracranial Electroencephalogram, Epilepsy, Seizure Prediction

Agradecimentos

Gostaria de começar por agradecer aos meus pais, por todo o apoio prestado, por toda a confiança e compreensão que depositaram em mim.

Queria agradecer à minha orientadora a Professora Elsa Ferreira Gomes, pela sua disponibilidade durante a realização deste projeto, pela ajuda prestada, bem como pela dedicação e empenho ao mesmo. Um especial agradecimento pela ajuda na elaboração deste documento.

E por fim agradeço a todos os amigos, em particular ao Rui Monteiro, meu colega de grupo e turma durante todo o mestrado.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

EEG	Eletroencefalograma
iEEG	Eletroencefalograma intracraniano
SE	Estado epilético
CSE	Estado epilético convulsivo
IA	Inteligência Artificial
CPU	Unidade de processamento central
GPU	Unidade de processamento gráfico
IDE	Ambiente de desenvolvimento integrado
SVM	Máquina de suporte de vetores
CSV	Valores separados por virgula
ROC	Características operacionais do recetor
AUC	Valor da área sob a curva ROC

1 Introdução

Neste primeiro capítulo é apresentado, de uma forma breve e resumida, o trabalho desenvolvido neste projeto. É feita uma descrição do conceito que originou o projeto através da descrição do problema, dos objetivos a atingir, bem como da análise de valor e dos resultados esperados.

1.1 Enquadramento

Este trabalho tem como foco principal a análise de dados obtidos por eletroencefalogramas (EEG), que estão normalmente associados ao registo da informação necessária, para o estudo da epilepsia.

A epilepsia, consiste na ocorrência de uma ou mais crises convulsivas, não relacionadas com problemas do foro cardíaco, alcoólico ou hipoglicémico. Por vezes, não é necessário a ocorrência de mais do que um episódio convulsivo para que se possa diagnosticar como sendo epilepsia, basta apenas que exista um elevado risco de outras possíveis ocorrências (CUF 2017).

O eletroencefalograma (EEG) permite registar a informação dos sinais elétricos das mais diversas áreas cerebrais. Normalmente a informação registada nos EEG, é recolhida através de elétrodos colocados no couro cabeludo, embora existam outros métodos menos comuns, como por exemplo elétrodos colocados na região subdural (meninges) ou córtex cerebral. (CUF 2017).

Apesar do EEG ser muito utilizado na deteção de convulsões, em casos como a epilepsia do lobo ultratemporal e lobo temporal (não lesional), não é suficiente. Nestes casos é necessário obter um registo de EEG mais próximo da origem do sinal. Para tal, usa-se um processo mais invasivo (geralmente necessitando de intervenção cirúrgica), colocando elétrodos na superfície do cérebro. O EEG resultante é conhecido por eletroencefalografia invasiva, o iEEG (Aashit et al 2014).

Estes dois conceitos aqui apresentados serão utilizados ao longo deste trabalho, e são pontos importantes relacionados com este tema.

1.2 Problema

Através da análise dos dados obtidos por eletroencefalogramas intracranianos, pretende-se detetar e prever convulsões em pacientes humanos.

O principal objetivo é a identificação dos períodos de maior probabilidade de ocorrências de convulsões, permitindo dessa forma uma deteção e previsão mais eficaz por partes dos sistemas baseados em iEEG. Se com a análise dos iEEG fosse possível a deteção destes períodos, os pacientes apenas teriam que se medicar em situações de convulsões iminentes, resultando numa redução geral dos efeitos secundários associados a este tipo de medicação.

1.3 Objetivos

O objetivo deste trabalho é, desenvolver uma metodologia usando algoritmos de *deep learning* capaz de classificar sinais resultantes do iEEG, para deteção e previsão de convulsões epiléticas em humanos.

Pretende-se ainda que seja elaborado um estudo sobre as diferentes técnicas utilizadas por outros autores, de forma a analisar e perceber se existe a possibilidade de aplicação de algumas das metodologias já utilizadas e comparar os resultados obtidos.

1.4 Resultados esperados

Como foi já referido, pretende-se construir uma metodologia capaz de classificar os sinais resultantes do iEEG, para deteção de convulsões epiléticas. Para tal, serão aplicadas técnicas para processamento e classificação dos sinais obtidos via iEEG.

Como resultado deste trabalho, espera-se que o sistema desenvolvido funcione de forma eficaz, identificando com segurança os períodos de maior probabilidade de ocorrência de convulsão. Para aferir essa segurança serão usadas métricas como taxa de acerto(*accuracy*), a *F-Measure* e AUC, a descrição destas medidas será abordada na secção 2.2.4, usando para teste, os dados disponibilizados pelo concurso Kaggle (Kaggle AM 2016). Espera-se que esta abordagem, usando *deep learning,* quando comparada com resultados já publicados para o mesmo *dataset*, obtenha melhores resultados.

Sendo o objetivo deste trabalho desenvolver uma metodologia capaz de detetar e prever convulsões a partir de iEEG e sabendo que a epilepsia afeta uma grande fatia da população mundial (Cook et al 2013), podemos dizer que tem valor de negócio. Esta aplicação, poderá ser utilizada por técnicos de saúde permitindo uma melhoria da qualidade de vida dos pacientes. Este assunto é tratado de forma mais detalhada na secção 3.3.

1.5 Abordagem

Neste trabalho, iremos criar uma aplicação de software, para previsão e deteção de convulsões epiléticas, com base em informação contida no IEEG.

Na sequência de trabalhos desenvolvidos anteriormente, pensou-se utilizar uma metodologia de deteção de padrões para extração de atributos (implementada na linguagem Java) e algoritmos de classificação como o *Random Forest*. Essa abordagem tinha já sido utilizada em deteção de sons cardíacos (Gomes et al, 2014).

Porém, ao aprofundar a pesquisa bibliográfica pensou-se em aplicar algoritmos de *deep learning*, em particular utilizar *Convolutional Networks*, usando o Tensorflow e a linguagem Python.

O ***Deep Learning***, é uma nova área de investigação de *machine learning* que tem tido uma grande evolução. Utiliza diversas camadas de redes neuronais profundas, para que lhe seja possível diferentes níveis de representação e abstração. Tem como principais objetivos o auxílio na resolução de múltiplos problemas, tais como reconhecimento de imagem, reconhecimento de voz, processamento de linguagem natural, ciências da vida e assistência à condução (Nvidia 2017).

De acordo com o estudo (Piotr et al 2008), *Convolutional Networks*, são sistemas utilizados para a extração e classificação de padrões altamente dimensionais tendo por base imagens ou variações múltiplas de tempo. São conhecidos como redes neuronais multicamada, que podem aprender através de funcionalidades de baixo nível e representações alto nível. Tem como principal vantagem, a construção de representações mais consistentes às alterações de tempo.

A escolha das tecnologias é fundamentada no subcapítulo 4.1.

1.6 Estrutura do documento

Nesta secção descreve-se brevemente os sete principais capítulos que compõem este documento.

Introdução: O capítulo da Introdução permite fazer a apresentação do projeto em questão, através da descrição do problema e o seu enquadramento, dos seus objetivos, dos resultados que se esperam atingir, da abordagem que se pretende utilizar, da análise de valor e da forma como o relatório foi organizado.

Machine Learning: O capítulo de *Machine Learning*, tem como principal objetivo descrever quais as metodologias de classificação existentes, abordadas nos estudos presentes no subcapítulo 3.2.1, assim como descrever os conceitos de aprendizagem supervisionada e *deep learning*.

Problema: O capítulo que descreve o Problema serve para apresentar de forma mais detalhada, o enquadramento do problema, a análise de valor e o estado de arte que irá apresentar e documentar soluções já existentes no mercado.

Bibliotecas *Deep Learning*: Este capítulo apresenta um pequeno estudo comparativo entre tecnologias Java e Python, a nível de desenvolvimento de aplicações orientadas para o *deep learning*. Este estudo irá comparar algumas das *frameworks* mais conhecidas neste âmbito e determinar qual a que obteve o melhor desempenho, segundo as métricas de comparação definidas.

Solução: Este capítulo, tem como principal objetivo fundamentar as escolhas das tecnologias utilizadas no desenvolvimento, especificar essas mesmas tecnologias e dar a conhecer todos os requisitos do projeto, assim como apresentar toda a documentação necessária e respetiva implementação dos mesmos.

Avaliação: Neste capítulo, são descritas as metodologias de avaliação utilizadas e serão apresentados os resultados obtidos.

Conclusões: No capítulo das Conclusões é apresentado o nível de sucesso de concretização dos requisitos propostos para o projeto e ainda possíveis limitações da solução. Serão descritos também alguns possíveis trabalhos que possam ser implementados no futuro. Por último, é feita uma apreciação final relativamente a este projeto.

2 *Machine Learning*

Os algoritmos de aprendizagem automática são usualmente catalogados como algoritmos de aprendizagem supervisionada, onde cada exemplo do *dataset* é associado a uma classe por uma etiqueta (ou categoria) e algoritmos de aprendizagem não supervisionada.

2.1 Aprendizagem Supervisionada

A aprendizagem supervionada é feita a partir de exemplos em que o *dataset* contém, para cada um dos seus exemplos, uma categoria associada. O modelo é construído através da definição das classes e dos exemplos de cada classe. Estes algoritmos conseguem aprender a classificar os diferentes casos em estudo nas diferentes categorias associadas (Goodfellow et al 2016).

Na solução implementada, a aplicação deste conceito, consistiu na criação de um vetor de etiquetas, em que cada posição deste vetor, correspondia à posição do ficheiro contido no *dataset* do caso em estudo. Esse vetor é composto por as 2 *labels*, a *label*, '0', significa que o sinal no ficheiro não contém crise convulsiva e a *label* '1', significa que o sinal gravado no ficheiro contém crise convulsiva.

2.2 Classificação

Neste subcapítulo, iremos detalhar algumas das metodologias mais utilizadas nos estudos apresentados no subcapítulo 3.2.1. As metodologias serão agrupadas nos seguintes subcapítulos: Pré processamento, Extração de atributos, Algoritmos de Classificação, Medidas de avaliação e Funções de ativação e de treino.

2.2.1 Pré-processamento

Atualmente torna-se complicado, para os algoritmos de classificação, conseguir extrair informação importante de volumes de informação tão grandes. No pré-processamento dos dados deve efetuar-se a remoção da informação indiscriminada deixando apenas a informação relevante (Dornhege et al 2007).

Os sinais EEG, podem conter diversos tipos de informação, como por exemplo: interferências de equipamentos eletrónicos, atividade muscular e ocular. Esses dados se não forem previamente removidos, podem influenciar de forma negativa os resultados obtidos (Dauwels et al).

O pré-processamento de sinais, pode ser utilizado aplicando diversos filtros para reduzir ruídos, diminuição da amostra, entre outros. Alguns exemplos de pré-processamento de sinais existentes, podem ser *Dataset Augmentation, Contrast Normalization* e *Downsampling* (Goodfellow et al 2016).

2.2.2 Extração de atributos

A extração de atributos, que costuma seguir-se à fase de pré-processamento dos dados, tem por objetivo identificar os atributos presentes no conjunto de dados. A extração de atributos pode resultar num grande volume de informação, levando à necessidade de utilização de técnicas de redução de dimensões, anterior ao processo de classificação. Com o aparecimento de algoritmos de aprendizagem máquina mais avançados, como por exemplo as CNNs, o *bottleneck* criado por este conceito pode ser facilmente resolvido e ultrapassado (Nurse et al 2016).

No subcapítulo 3.2.1, onde são apresentados alguns trabalhos sobre detecção e previsão de convulsões existentes na bibliografia, são referidos alguns métodos de extração de atributos, nomeadamente:

O ***Fast Fourier Transform*** (FFT), consiste na aplicação de ferramentas matemáticas para a análise do sinal (como o EEG). As características do sinal a analisar, são trabalhadas, de forma a que seja possível uma representação seletiva da amostra desse sinal, utilizando a densidade de poder espectral (PSD) (Al-Fahoum et al 2014).

O ***Wavelet Transform*** (WT), tem como principal objetivo, a compressão do sinal (no caso, biomédico) em pequenos parâmetros que representam o sinal. Esta forma de transformação subdivide-se em *Continuous Wavelet Transform* (CWT) e *Discrete Wavelet Transform* (DWT) (Al-Fahoum et al 2014).

2.2.3 Algoritmos de Classificação

Neste subcapítulo, pretendemos descrever alguns dos algoritmos de classificação, utilizados no 3.2.1.

O ***Support Vector Machine*** (SVM), é um algoritmo de classificação linear, que faz a distinção entre classes utilizando funções lineares e que utiliza para a identificação de classes, *hyperplanes* discriminativas. Permite também a criação de limites de decisão não lineares. Este algoritmo é conhecido pelo sucesso na resolução de problemas associados à computação cerebral e tens diversas vantagens tais como, propriedades de generalização e poucas ou nenhumas limitações em questões de excesso de treino (Lotte et al 2007).

As ***Artificial Neural Networks*** (ANN), consistem em sistemas computacionais distribuídos compostos por neurónios artificiais (ou unidades de processamento), densamente interligadas, que funcionam como um todo para resolver um problema (Lotte et al 2007). Os neurónios estão dispostos em camadas com um elevado número de ligações. Estas ligações, possuem pesos associados na rede. Os valores destes pesos são ajustados no processo de aprendizagem da rede. Os neurónios, recebem os valores que são ponderados e combinados por uma função matemática, a função de ativação (Stanford NN 2017) (Gama et al 2012).

Uma ANN é caracterizada pela sua arquitetura e aprendizagem. Sendo que, a primeira, diz respeito ao número de neurónios (e a forma como estão ligados) e a segunda às regras de ajuste dos pesos da rede. A ANN pode conter mais do que uma camada de neurónios (multicamadas). As ligações, podem apresentar diferentes padrões de ligação. Assim, pode ter-se uma rede ligada completamente (os neurónios estão todos ligados entre camadas), parcialmente ou localmente (apenas ligados numa região). As redes podem apresentar ligações de retroalimentação (*feedback*) que permitem que um neurónio receba como entrada, a saída de um neurónio da mesma camada ou da camada posterior. As redes mais comuns são as *feedforward*, de alimentação progressiva (Gama et al 2012).

O ***K Nearest Neighbors*** (kNN), tem por base um conjunto de treino e o seu principal objetivo é a atribuição de uma categoria, a uma classe predominante, contida nos pontos k pertencentes ao conjunto de treino, são normalmente eficientes quando utilizados com *feature vectors* de baixas dimensões (Lotte et al 2007).

O ***Random Forest***, consiste na construção de árvores de decisão, para a classificação e regressão. Nos problemas de classificação, as árvores tendem a votar na classe mais popular, nos problemas de regressão, as suas respostas são ponderadas de forma a poder obter uma estimativa. Tem como principais vantagens, a taxa de acerto, a sua eficiência de execução em bases de dados com muita informação, a forma de lidar com milhões de variáveis de introdução, sem as eliminar, entre outras (DataScienceCentral 2017) (Statistica 2017).

A **Regressão logística**, é um modelo de regressão múltipla, mas com uma variável de resposta categórica dicotómica (dependente) e variáveis preditivas (independentes) podem ser contínuas ou categóricas. Quando a variável dependente é dicotómica (Verdadeiro/Falso ou 0/1), não pode ser usada a regressão linear (Peng et al 2002).

As ***Convolutional Networks,*** foram desenhadas especialmente para a extração e classificação de padrões multidimensionais a partir de imagens ou series de tempo multivariável. São redes não lineares, multicamada e que podem ser treinadas. A forma de treinar as suas camadas, ocorre em simultâneo através dos algoritmos de aprendizagem designados de *back-propagation*, dessa forma as camadas podem aprender funcionalidades de baixo nível e representações de alto nível de uma forma integrada (Mirowski et al 2008).

2.2.4 Medidas de avaliação

Com este subcapítulo, pretendemos descrever as medidas de avaliação utilizadas nos estudos abordados no 3.2.1.

Para a utilização das medidas de avaliação, é necessário perceber que existem medições comuns, que permitem caracterizar o desempenho. Estas medições são utilizadas por cada uma das medidas e serão explicadas abaixo (por simplicidade, para o caso de duas classes) (Casson et al) (Gama et al 2012).

- Verdadeiro Positivo(VP), representa o número de exemplos da classe positiva classificados corretamente. Neste caso, a correta deteção de convulsões;
- Falso Positivo(FP), representa o número de exemplos da classe negativa, mas classificados como sendo da classe positiva. Neste caso, a não deteção de convulsões, quando o são;
- Verdadeiro Negativo(VN), representa o número de exemplos da classe negativa classificados corretamente. Neste caso, a correta deteção de “não convulsões”;
- Falso Negativo (FN), representa o número de exemplos da classe positiva classificados incorretamente, como da classe negativa. Neste caso, a deteção de convulsões, quando não o são;
- Taxa Falsos Positivos (TFP), correspondendo à taxa de acerto na classe negativa;
- Taxa Verdadeiros Positivos (TVP), correspondendo à taxa de acerto na classe positiva.

Estas medições são aplicadas em fórmulas para cada uma das medidas abaixo.

2.2.4.1 Sensibilidade

A Sensibilidade, corresponde à taxa de acerto na classe positiva. Esta medida de avaliação, aplicada ao caso em estudo, consiste na probabilidade da antecipação de uma convulsão, num determinado período de tempo e é dos pontos mais importantes na avaliação de algoritmos de previsão (Snyder et al 2008).

E que pode ser calculada através de (Casson et al) (Gama et al 2012):

$$\frac{VP}{VP + FN}$$

2.2.4.2 Especificidade

A Especificidade, corresponde à taxa de acerto na classe negativa. Pode ser definida como, a probabilidade de conseguir determinar a qualquer momento, um estado de pré convulsão (Snyder et al 2008).

E pode ser calculada através de (Casson et al) (Gama et al 2012):

$$\frac{VN}{VN + FP}$$

2.2.4.3 Taxa de acerto

Pode ser calculada através de (Gama et al 2012):

$$\frac{VP + VN}{VN + VN + FP + FN}$$

2.2.4.4 *Precision*

Pode ser calculada através de (Casson et al) (Gama et al 2012):

$$\frac{VP}{VP + FP}$$

2.2.4.5 *Recall*

Pode ser calculada através de (Casson et al) (Gama et al 2012):

$$\frac{VP}{VP + FN}$$

2.2.4.6 *Area under the curve* (AUC) / Curva ROC

O gráfico ROC é um gráfico bidimensional num espaço (ROC), com dois eixos que representa as medidas de TFP e TVP. A área abaixo da curva ROC designa-se por AUC (*Area Under Curve*) e varia entre 0 e 1, sendo que os valores mais próximos de 1 são considerados melhores (Gama et al 2012). Num problema de binário (duas classes) pode usar-se as curvas ROC para análise do desempenho. Na Figura 1, temos um exemplo de uma curva ROC.

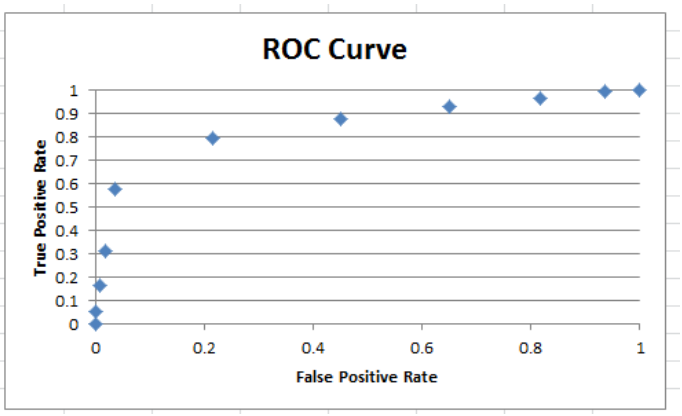

Figura 1 - Curva ROC

2.2.4.7 *F-Measure*

A medida F (*F-Measure*), é a média harmónica ponderada da taxa de acerto com a taxa de verdadeiros positivos. Esta medida, combina as medidas *Precision* e *Recall* (Gama et al 2012).

2.2.5 Funções de ativação

Neste subcapítulo iremos descrever as várias funções de ativação, utilizadas no 5.3.

2.2.5.1 Relu

A função de ativação linear retificada, Relu (*rectified linear unit*), é definida pela fórmula matemática $f(x) = max(0, x)$ e consiste na ativação de valores muito próximos de zero (CS231n NN 2017). Esta função de ativação é a mais recomendada para utilizar na maioria das redes neurais *feedforward*. A aplicação desta função, à saída de uma transformação linear, produz uma transformação não linear, mas muito próxima da linear (com duas partes lineares), como se pode observar na Figura 2 (Goodfellow et al 2016).

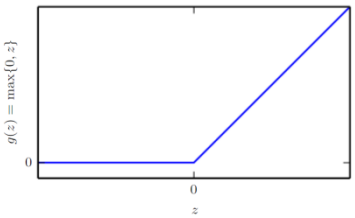

Figura 2 - Função de ativação linear retificada (Goodfellow et al 2016)

Como as unidades lineares retificadas são quase lineares, preservam muitas das propriedades que tornam os modelos lineares fáceis de otimizar com métodos baseados em gradientes (Goodfellow et al 2016).

2.2.5.2 Softplus

A função Softplus, é uma aproximação suave da função retificada (ver Figura 3) e é definida pela expressão matemática: $log(1 + exp(x))$ (Goodfellow et al 2016).

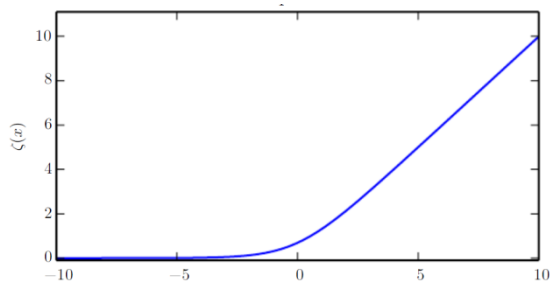

Figura 3 - Função Softplus (Goodfellow et al 2016)

2.2.5.3 Sigmoid

A função logística Sigmoid, ver Figura 4, pode ser calculada através da formula matemática $\sigma(x) = 1/(1 + exp(-x))$ e consiste transformação de um valor real num intervalo de 0 e 1, ou seja, número negativo relativamente grande é transformado em 0 e um positivo é transformado em 1 (CS231n NN 2017) (Goodfellow et al 2016).

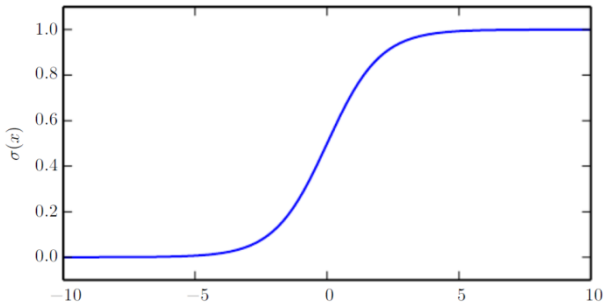

Figura 4 - Função Sigmoid (Goodfellow et al 2016)

2.2.5.4 Tanh

Pode ser calculada através da formula matemática $tanh(x) = 2\sigma(2x) - 1$ e, onde σ representa a função de ativação logística sigmoid, e consiste na transformação de um valor real num intervalo de -1 e 1 (CS231n NN 2017). Segundo Goodfellow, quando uma função de ativação sigmoidal deve ser usada, a função tangente hiperbólica, em geral, funciona melhor do que a função sigmoide logística (Goodfellow et al 2016).

2.2.6 Funções de treino

Neste subcapítulo iremos descrever as várias funções de treino, referidas no 5.3, nomeadamente Gradient descent e Adam optimizer. Estas funções são algoritmos de otimização que se utilizam para encontrar os melhores parâmetros dos modelos de *deep learning*. Muitos dos algoritmos de *deep learning*, envolve otimização.

2.2.6.1 Gradient descent

É um algoritmo de otimização que consiste na procura de valores que serão utilizados como parâmetros (coeficientes) de uma função (MachineLearningMistery GRD DSC 2017). A otimização envolve a tarefa de maximizar ou minimizar uma função, a função objetivo. Na Figura 5, pode observar-se como a derivada de uma função pode ser usada para encontrar um mínimo. Esta tecnica é conhecida por Gradient descent (Goodfellow et al 2016).

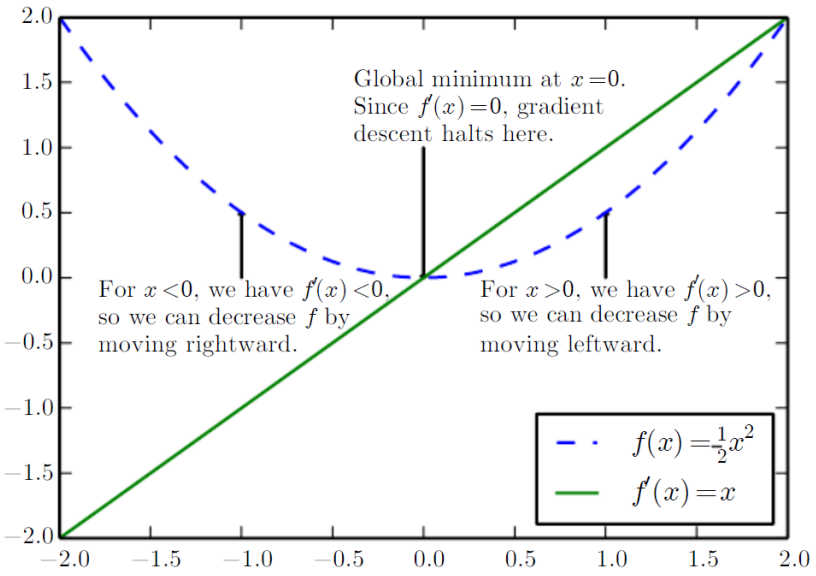

Figura 5 - Função Gradiemt descent (Goodfellow et al 2016).

2.2.6.2 Adam optimizer

É um algoritmo de otimização, que através dos dados de treino, calcula e atualiza os pesos da rede, é também um algoritmo fácil de configurar e que garante bons resultados. O nome "Adam" deriva da frase "*adaptive moments*." (MachineLearningMistery ADM 2017).

2.3 *Deep learning*

O *deep learning*, é um tipo de aprendizagem automática que permite que os sistemas melhorem com a experiência e os dados. Este tipo de aprendizagem possuí grande poder e flexibilidade pela forma como aprende a representar, baseado em hierarquias de conceitos, em que cada conceito é definido relativamente a uns mais simples, e os mais abstratos são definidos relativamente aos menos abstratos (Goodfellow et al 2016).

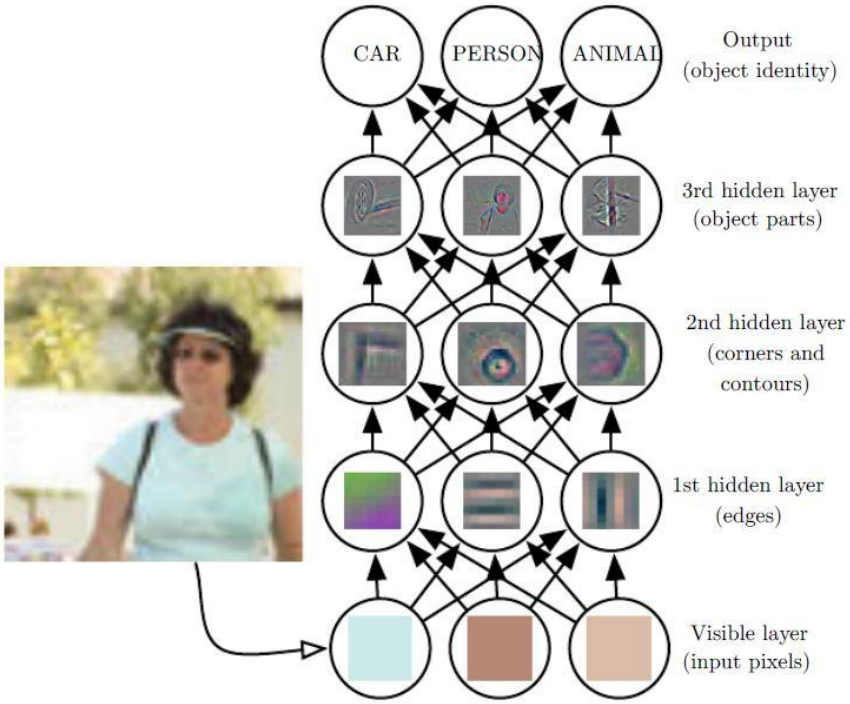

Figura 6 - Representação de um modelo *deep learning* (Goodfellow et al 2016)

A Figura 6, representa um exemplo de um modelo de *deep learning*. Esse modelo divide-se em várias *layers*, nomeadamente: a *Visible layer*, que representa a camada onde são mantidas as imagens introduzidas na rede sobre a forma de *arrays* de pixéis e as *Hidden layers*, que permitem a extração de diversas *features* abstratas das imagens, os seus valores não estão visíveis e cabe ao modelo, encontrar na informação observada, os conceitos mais relevantes (Goodfellow et al 2016).

3 Problema

Este trabalho, surge a partir de um desafio lançado pelo Kaggle (Kaggle 2016) e que visa desenvolver uma metodologia capaz de classificar sinais resultantes de iEEG para detetar e prever convulsões epiléticas em humanos. O desafio consiste em distinguir, em gravações de dez minutos, os períodos que antecedem uma convulsão e os dez minutos que correspondem a gravações iEEG com atividade *interictal* (intervalo entre as convulsões).

Neste capítulo, para além do contexto no qual se desenvolveu o projeto, apresenta-se o problema a resolver, os conceitos essenciais relevantes para a compreensão do trabalho realizado. É feita uma da análise de valor do projeto detalhada, assim como um estado de arte.

3.1 Enquadramento do Problema

A epilepsia é uma doença neurológica, que afeta entre 4 a 7 mil habitantes em Portugal, é a quarta doença mais comum nos Estados Unidos e afeta mundialmente 65 milhões de pessoas. A probabilidade de ocorrência de uma crise convulsiva, em pessoas não epiléticas, é de 1 em cada 20 (CUF 2017).

As convulsões, consistem na alteração da atividade elétrica cerebral e podem não ter nenhuma explicação lógica, ou podem ter causas associadas a histórico familiar. As crises são normalmente de curta duração, repetem-se ao longo do tempo e a sua frequência difere de pessoa para pessoa (CUF 2017).

As convulsões são compostas por diversas fases, que podem ser categorizados como:

- *preictal*: Fase anterior à crise convulsiva, pode ou não despoletar alguns sintomas, e caso os despolete, podem ter duração indefinida (EpilepsyColorado 2017);
- *ictal*: Fase da crise convulsiva (EpilepsyColorado 2017);

- *interictal*: Fase entre crises, em alguns casos podem ocorrer perturbações emocionais (EpilepsyColorado 2017);
- *posIctal*: Fase final, em que dependendo do tipo e da intensidade da convulsão, pode levar a diferentes tempos de recuperação (EpilepsyColorado 2017).

3.2 Estado de arte

No contexto deste trabalho, foram analisadas algumas abordagens já existentes para deteção e previsão de ataques de epilepsia, assim como métodos de avaliação utilizados, que serão abaixo documentados e apresentados com algum detalhe de forma a permitir uma perceção sobre os avanços já existentes nesta área.

3.2.1 Soluções existentes para deteção e previsão de convulsões

Neste subcapítulo, apresentam-se as abordagens estudadas para este trabalho. Na Tabela 1, apresenta-se de forma resumida, os principais pontos chave utilizados em cada um dos estudos, desde medidas de avaliação, metodologias, *dataset* e os testes utilizados.

Tabela 1 - Soluções existentes

Referências	*Dataset*	Medidas de Avaliação	Metodologias utilizadas	Teste estatístico
(Cook et al 2013)	15 pacientes, equipamentos implantados cirurgicamente (intracraniano)	Sensibilidade. Coeficiente de correlação de Spearman.	Correlação. *Leave-one-out cross validation*.	Teste correlação entre eventos reportados e eventos gravados
(Brinkmann et al 2016)	2 humanos e 8 cães, equipamentos implantados cirurgicamente	AUC; curvas ROC, Sensibilidade se Especificidade a 75%	Transformada *Fast Fourier*; janela de Hamming; 3 *fold cross validation*; *Random Forest;Bagged SVM;LassoGLM*; SVM; Algoritmos genéticos;CNN;	Testes de hipóteses, método de Hanley Mcneil
(Gadhoumi et al 2015)	Freiburg *dataset*	ROC; AUC	Hilbert Transform; *leave one out; five fold cross validation*; SVM	
(Karoly et al 2016)	15 pacientes (com 16 canais)	Coeficiente de correlação de Spearman; Sensibilidade.	Correlação; Template Matching; Cross correlation.	Teste Wilcoxon rank sum; teste Shapiro-Wilk
(Snyder et al 2008)	4 Pacientes	Sensibilidade e Especificidade;	Análise de EEG e alerta em tempo real. Depois do pré-processamento do sinal, aplica algoritmo de	Teste emparelhado à igualdade das medianas da sensibilidade do

Referências	*Dataset*	Medidas de Avaliação	Metodologias utilizadas	Teste estatístico
			classificação kNN.	algoritmo. Simulação Monte Carlo para gerar previsões aleatórias.
(Haut et al 1999)	76 pacientes, entrevistas		Análise univariada com correção Mantel- Haenszel	Teste de Fischer
(Mirowski et al 2009)	21 pacientes Freiburg *dataset*	Sensibilidade e Taxa de Falsos Positivos (TFP)	Extração de atributos bivariados calculados em janelas temporais de 5 seg (6 tipos); *Wavelet Transformation*; *Cross-validation*; SVM, Regressão Logística e CNN;	
(Kevrik et al 2012)	21 pacientes	Taxa de acerto;	DWT e PCA (MSPCA) para pré-processamento, DWT para extração de atributos; 10 fold cross validation	
(Mirowski et al 2008)	21 pacientes Freiburg *dataset*		Extração de atributos bivariados (janelas temporais de 5 seg); *Wavelet*; *Convolutional Networks*, SVM, Regressão Logistica	
(Park et al 2011)	21 pacientes Freiburg *dataset*	Sensibilidade	Atributos são calculados a partir da potência espectral em 9 faixas em janelas de 20 seg do sinal. Filtro de Kalman; 5 *fold cross validation*; SVM; *Cost sensitive* SVM (CSVM);	
(Schindler et al 2007)	60 pacientes		Correlação.	
(Antoniades et al 2016)	25 Pacientes	Taxa de acerto	Extração de atributos do sinal EEG. *Deep learning*, *Convolutional Neural Networks*	Teste de McNemar
(Page et al 2014)	9 pacientes	Taxa de acerto, F-*Measure*, Complexidade e memória computacional	9 atributos, retirados do sinal EEG (eg, energia média); SVM, kNN, Reg. Logistica, Deep NN: SdA e DBN	

Dos trabalhos apresentados na Tabela 1, é possível observar que embora as abordagens seguidas nos vários trabalhos sejam pouco semelhantes, destaca-se, como sendo as mais utilizadas nos diversos estudos, a utilização da taxa de acerto, como medida de avaliação, o SVM como algoritmo de avaliação e o *Random Forest, Convolutional Neural Networks* e a Regressão Logística como algoritmos de classificação. Estas e outras metodologias utilizadas, serão explicadas em maior detalhe nos subcapítulos 2.2.1, 2.2.2, 2.2.3, 2.2.4.

Destes estudos destaca-se principalmente o (Brinkmann et al 2016). Este estudo teve por base o concurso do Kaggle (Kaggle AM 2016) e que teve como principal objetivo, documentar as soluções e metodologias utilizadas pelas equipas com melhor classificação.

Este concurso foi composto por várias equipas, que implementaram uma solução de previsão e deteção de convulsões, com base nos *datasets* de iEEG fornecidos pelo concurso. As metodologias utilizadas por cada uma das equipas, que aceitaram descrever a sua forma de implementação, serão descritas na Tabela 2. Foram utilizadas como medidas de avaliação, a *Area under the curve* (AUC) e Curvas (ROC).

Estas equipas utilizaram algumas metodologias bastante conhecidas nesta área e algumas delas, também serão utilizadas no desenvolvimento no nosso trabalho.

Tabela 2 - Metodologias utilizadas pelas equipas

	Metodologias utilizadas	**Linguagem**
1º Equipa	3 Algoritmos utilizados • 1º *LassoGLM* (1/2); • 2º *Fast Fourier* Transform e SVM (1/4); • 3º *Random forest* com 80 árvores (1/4). Média ponderada dos 3 algoritmos para obter o resultado final.	Python
2º Equipa	*Downsampling*, *Fast Fourier Transform,* SVM e *Platt scaling.*	Python
3º Equipa	Hamming Window, *Fast Fourier Transform*, Bayesian model e kNN.	R
4º Equipa	Algoritmo genético, 3 *fold cross validation*, SVM.	
8º Equipa	*Downsampling*, *Fast Fourier Transformation*, GLMNet, SVM.	R
9º Equipa	*Convolutional Neural Networks*.	

A escolha deste estudo, surgiu pelo facto, de realizar a comparação entre diversas equipas que desenvolveram algoritmos de previsão e deteção de convulsões baseados em sinais EEG, fornecidos pelo concurso atual. Essas soluções poderão também servir para estabelecer comparações com a solução que iremos implementar, uma vez que também iremos utilizar o mesmo *dataset* de sinais iEEG.

3.3 Análise de Valor

Neste subcapítulo, serão apresentados alguns dos conceitos abordados no módulo de competências, Analise de Valor, e tem como principal objetivo enquadrar esses mesmos conceitos neste projeto.

3.3.1 Novo modelo de desenvolvimento de conceito(NCD)

O novo modelo de desenvolvimento de conceito NCD, define cinco pontos chave: a identificação e análise da oportunidade, a seleção e criação da ideia e a definição do conceito.

3.3.1.1 Identificação da Oportunidade

Sendo a epilepsia, uma doença que não tem cura e que afeta mundialmente mais de 65 milhões de pessoas, considera-se que as formas atuais de tratamento, para 30 a 40% dos casos não é adequada ou não estão disponíveis para estes pacientes (Cook et al 2013).

Este elemento chave pode ser analisado através dos seguintes métodos, definição de um *Roadmapping*, a análise das tendências tecnológicas e dos consumidores, análise de inteligência competitiva, investigações de mercado e planeamento de cenários (Koen et al).

3.3.1.2 Analise da Oportunidade

Assim, para combater a falta de eficácia dos tratamentos existentes e de forma a melhorar a qualidade de vida dos pacientes, foram sendo pensadas novas formas de tratar/controlar a doença, tendo começado a surgir sistemas capazes de prever e detetar atempadamente os períodos prévios à ocorrência de convulsões, permitindo uma atuação imediatamente antes da ocorrência, reduzindo drasticamente os possíveis efeitos secundários associados (Cook et al 2013).

Este subcapítulo, pode ser analisado através do mesmo método utilizado para encontrar oportunidades futuras, sendo que o esforço associado teria que ser muito mais detalhado e através da atribuição a um longo projeto, uma equipa multifuncional de 3 a 5 elementos a tempo inteiro (Koen et al).

3.3.1.3 Criação da Ideia

Existem já, diversos estudos que têm como objetivo criar estes sistemas de previsão e deteção de convulsões, mas que recorrem a outras metodologias para o processamento e análise da informação. Após a análise de alguns dos estudos existentes, foi possível perceber que estes, apresentavam resultados favoráveis, mas ainda fracos comparativamente aos que utilizavam *deep learning*.

Alguns dos métodos possíveis para a análise deste ponto chave são: abordagens etnográficas, identificação de novas soluções tecnológicas, a existência de uma cultura organizacional que permita aos os seus empregados, testar e validar as ideias, que não sejam apenas as suas, a existência de muitos incentivos para proporcionar a estimulação de ideias e incluir pessoas com diferentes estilos cognitivos na equipas de enriquecimento de ideias (Koen et al).

3.3.1.4 Seleção da ideia

A nossa solução, prevê a utilização de uma abordagem, ainda pouco utilizada neste tipo de aplicações e consiste na técnica de *deep learning*, recorrendo ás tecnologias documentadas no subcapítulo 4.1.1.

O ponto chave seleção da ideia, pode ser analisado através dos métodos, de metodologias de portefólios, baseados em múltiplos fatores(não apenas com justificação financeira), utilizando *anchored scales*, processos de seleção de ideias com a passagem do feeback imediato até aos criadores das ideias e a utilização de opções teóricas na avaliação dos projetos (Koen et al).

3.3.1.5 Definição do Conceito

Desenvolvimento de uma aplicação de software, que tem como principal objetivo a deteção e previsão de períodos de maior probabilidade de ocorrência de convulsões, baseado na análise de iEEG, utilizando *deep learning*, mais concretamente redes neuronais convolucionais para processamento da informação. Esta aplicação, permitirá ao paciente controlar esta doença de forma mais eficaz e melhorar a sua qualidade de vida.

Alguns métodos que podem ser utilizados para analisar este elemento chave, podem ser a definição de critérios numa empresa que descrevam o que pode ser considerado um projeto atrativo, a evolução rápida de inovações com elevado potencial, envolvimento do cliente em testes reais ao produto e procurar abordagens cientificas alternativas (Koen et al).

3.3.2 Valor

Este conceito, pode ser definido de muitas e diferentes formas, e que de acordo com diferentes contextos teóricos onde se utilizou, foi definido como necessidade, desejo, interesse, critério, crenças, atitudes e preferências (Nicola et al 2012).

3.3.2.1 Valor para o cliente

Este conceito, representa a vantagem que surge entre a associação pessoal com a oferta organizacional. Pode ocorrer através da redução de sacrifícios, existência de benefícios, o resultado obtido através da combinação ponderada entre o sacrifício e o benefício (Woodall 2003).

3.3.2.2 Valor percetível

Este conceito, representa, de acordo com as necessidades e expetativas, as diferenças entre os benefícios e sacrifícios percebidas pelos clientes (Lapierre 2000).

No âmbito desta dissertação, a implementação do sistema de previsão e deteção de convulsões com base nas leituras iEEG, traz valor para os pacientes que irão utilizar um sistema semelhante, pois permitirá uma melhoria significativa na sua qualidade de vida, contém também valor para os programadores envolvidos, pois irá permitir adquirir novos conhecimentos sobre, a doença em questão, outros estudos já realizados e sobre as tecnologias utilizadas.

Na Tabela 3, é apresentada a perspetiva longitudinal do valor com os benefícios e sacrifícios para o cliente neste contexto.

Tabela 3 - Perspetiva longitudinal de valor

	Benefícios	**Sacrifícios**
Antes da Compra	Segurança; Menor risco; Qualidade de vida;	Preço.
Transação		Custos de aquisição; Financeiro.
Após a compra	Customização; Suporte.	Custos de entrega e instalação; Custos de reparação; Custos de manutenção.
Após a utilização	Previsão/deteção de convulsões em tempo real; Redução da medicação; Redução de efeitos secundários; Segurança; Fiabilidade; Menor risco; Qualidade de vida.	

3.3.3 Proposta de valor

Desenvolver um algoritmo de previsão e deteção de convulsões, com uma elevada taxa de acerto, para garantir a fiabilidade da previsão. Desta forma, proporcionar aos utilizadores, pacientes com epilepsia, um sistema de monitorização, que permitirá garantir a redução da medicação associada, redução dos efeitos secundários, melhoria significativa da qualidade de vida e um maior controlo da doença.

Não sendo a única aplicação do mercado, pretende destacar-se pela metodologia de desenvolvimento utilizada, para processamento dos sinais iEEG, de modo a conseguir detetar/prever convulsões rapidamente e com uma elevada taxa de acerto.

3.3.4 Modelo Canvas

Na Figura 7, podemos observar o modelo Canvas que foi elaborado para este projeto.

Figura 7 - Modelo Canvas

Com base na Figura 7, podemos concluir que:

Este software tem como parceiros todos os profissionais de saúde e respetivas instituições médicas.

Irá ser criado um software de previsão e deteção de convulsões, através da análise de iEEG de forma inovadora, destinado a pacientes com epilepsia. Desta forma, pretende-se melhorar a qualidade de vida dos pacientes proporcionando uma forma de tratamento mais eficaz, reduzindo a medicação e os custos associados.

O principal meio de venda são lojas especializadas ou profissionais de saúde e o valor de compra adequa-se à disponibilidade financeira do paciente.

4 Bibliotecas de *deep learning*

Este capítulo apresenta um estudo comparativo entre duas tecnologias e as suas *Framework* para o desenvolvimento de aplicações orientada ao *deep learning*, que permitirá suportar a escolha das linguagens utilizadas neste trabalho. Descreve também, quais as tecnologias utilizadas na implementação.

4.1 Design

Para linguagem de programação foram consideradas as linguagens Java e Python. Assim sendo, procedeu-se a um pequeno estudo comparativo entre as duas linguagens Python e Java e com base no trabalho de Kovalev (Kovalev et al 2016), foi possível comparar diferentes *Framework*, no desenvolvimento de software orientado ao *deep learning*, segundo diversas métricas e casos de teste, que serão detalhados nesta secção.

O **Java**, é uma tecnologia já bastante presente no mercado, com uma comunidade de desenvolvimento a nível mundial massiva, que tem vindo a melhorar significativamente ano após ano, e que se mantêm como a linguagem mais utilizada há vários anos. É uma tecnologia que permite o desenvolvimento em diversas áreas, mas que embora também se consiga utilizar nesta área de estudo, as *Framework* disponíveis para o efeito são poucas, tendo apenas encontrado uma, designada de DeepLearning4J (Java 2017).

Embora o **Python**, seja uma tecnologia que tem vindo a ganhar força ao longo dos anos, ainda precisa de mais maturidade para poder competir com o Java, e que para além de se poder utilizar nas mais diversas áreas também, ainda não tem o peso do seu concorrente. Relativamente à utilização do Python orientado ao *deep learning*, podemos observar que esta linguagem está mais vocacionada para tal, uma vez que tem a maior comunidade de desenvolvimento e possui algumas *Framework* de peso bem conhecidas, designadas de Theano e Tensorflow, contrariamente ao Java (Python 2017).

Podemos observar a hegemonia do Java, na Figura 8, conforme acima explicado, relativamente às outras linguagens.

Programming Language	2017	2012	2007	2002	1997	1992	1987
Java	1	1	1	2	17	-	-
C	2	2	2	1	1	1	1
C++	3	3	3	3	2	2	4
C#	4	4	6	10	-	-	-
Python	5	7	7	16	27	-	-
PHP	6	5	4	6	-	-	-
JavaScript	7	9	8	7	20	-	-
Visual Basic .NET	8	24	-	-	-	-	-
Perl	9	8	5	4	3	-	-
Assembly language	10	-	-	-	-	-	-
Lisp	28	12	12	9	7	11	2
Ada	30	15	16	15	10	3	13
Prolog	33	43	27	27	14	9	3

Figura 8 - Top anos anteriores linguagens programação (Tiobe 2017)

Para a comparação das *Framework* destas linguagens, recorreu-se ao estudo elaborado por (Kovalev et al 2016) que teve como principal objetivo a comparação entre as diferentes *Framework* orientadas ao *deep learning*, tais como Theano, Torch, Caffe, Tensorflow e DeepLearning4J.

Os testes foram realizados utilizando redes neuronais com variação de profundidade e largura, que podem ser observadas na Figura 9 e na Figura 10. Essas redes foram testadas segundo as funções de ativação, Tangente Hiperbólica (Tanh) e *Rectified linear unit function* (ReLU) e tiveram como termos comparativos para análise dos resultados obtidos nas diferentes fases de teste, os seguintes pontos (Kovalev et al 2016):

- Tempo de convergência:
- Tempo de previsão;
- Taxa de acerto;
- Tamanho do código fonte.

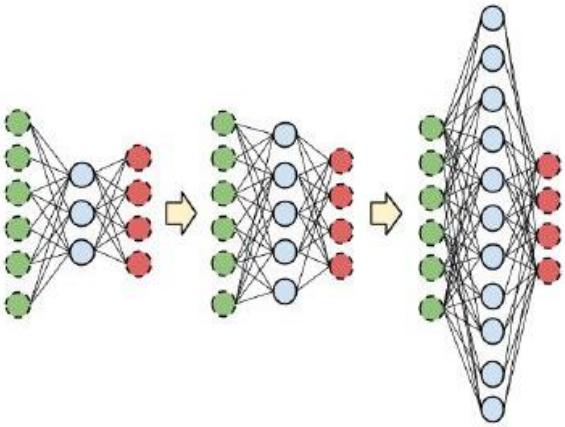

Figura 9 - Rede neuronal variação em largura (Kovalev et al 2016)

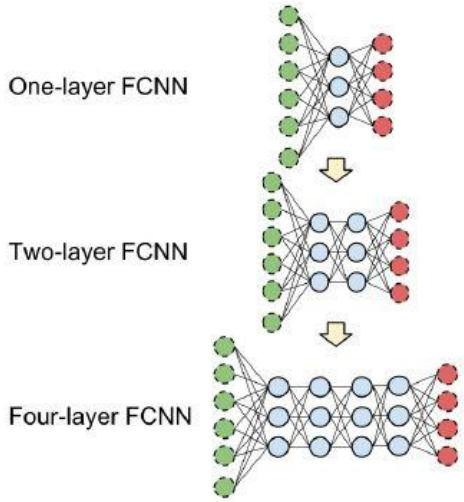

Figura 10 - Rede neuronal variação em profundidade (Kovalev et al 2016)

Com base nos resultados obtidos, nos testes realizados, e medindo a velocidade de processamento, a taxa de acerto e a complexidade do código, para cada uma das *Framework*, permitiu elaborar um ranking com as seguintes classificações (Kovalev et al 2016):

1. Theano (Python);
2. Tensorflow (Python);
3. Caffe (Protobuf);
4. Torch (Lua);
5. DeepLearning4J (Java).

Tendo em conta o ranking apresentado acima, consideramos que as duas melhores *Framework* são Theano e Tensorflow, da linguagem Python. Assim sendo iremos analisar as duas de forma a perceber qual a mais adequada para a aplicação pretendida.

O estudo efetuado por Al-Rfou et al (Al-Rfou et al 2016), teve como principal objetivo documentar o Theano e as suas funcionalidades. Dessa forma, foi possível utilizar alguma da informação presente, sobre testes realizados entre diferentes *Framework*, para complementar o estudo acima apresentado e auxiliar na tomada de decisão, sobre a melhor *Framework*, de Python, a utilizar.

Esses testes consistiram na comparação das *Framework* respetivas, em diferentes modelos de aprendizagem máquina, designados de *Convolutional Networks*, *recurrent neural networks* e *recurrent neural networks* para o mapeamento sequência a sequência, e teve como principal objetivo demonstrar o desempenho obtido em cada uma das fases para cada um dos modelos indicados (Al-Rfou et al 2016).

Na primeira fase de testes, foram medidos os desempenhos das *Framework,* para as *Convolutional Networks*, onde foram utilizados os modelos convolucionais, AlexNet [1], OverFeat[2], VGG [3]e GoogleLeNet V1, para processamento do *Imagenet dataset*. (Al-Rfou et al 2016).

Na segunda fase de testes, foram medidos os desempenhos das *Framework*, para *recurrent neural networks*, tendo-se utilizado variantes de modelos LSTM para processamento do *dataset Penn Treebank*. (Al-Rfou et al 2016).

Na terceira fase de testes, foram medidos os desempenhos das *Framework* para as *recurrent neural networks for sequence-to-sequence mapping.* (Al-Rfou et al 2016).

Em conclusão, os estudos acima apresentados mostram que o Python, é a linguagem mais forte no desenvolvimento do software orientado ao *deep learning*, tendo duas das melhores *Framework* para tal. Essas *Framework*, quando comparadas entre si, demonstram que sendo ambas atuais (sendo uma bem mais recente), e embora o Tensorflow se tenha destacado um pouco mais, ambas conseguem ter resultados similares no desempenho do processamento de informação, para os diferentes casos de teste realizados no referido trabalho (Al-Rfou et al 2016).

Assim, a escolha para este trabalho, recai sobre o Tensorflow, pelas qualidades já acima enumeradas, mas também por a sua utilização, estar associada a grandes empresas (Tensorflow 2017).

4.1.1 Tecnologias utilizadas.

O **Python** é uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, funcional, que permite uma integração de sistemas mais eficiente.

[1] https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet
[2] https://github.com/sermanet/OverFeat
[3] https://github.com/ry/tensorflow-vgg16

Esta linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Dá prioridade à legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos da sua biblioteca padrão e por módulos e *Framework* desenvolvidos por terceiros (Python 2017).

O **Tensorflow** é uma biblioteca *open source* para computação numérica utilizando grafos de fluxo de dados. Os nós nos grafos representam operações matemáticas, enquanto que as pontas do grafo representam vetores (*tensors*) de dados multidimensionais que comunicam entre si. A arquitetura flexível permite fazer o *deploy* da computação para um ou mais CPUs (ou GPUs) num computador, servidor, ou equipamento móvel como uma única API (Tensorflow 2017).

O **Pycharm**, software de desenvolvimento para a linguagem Python, pertencente à empresa JetBrains, é um IDE que permite a integração de diversas ferramentas científicas, tais como Anaconda, matplotlib e NumPy. Contém muitas ferramentas de desenvolvimento integradas e auxilia o desenvolvimento de forma inteligente, entre outras qualidades (Pycharm 2017).

Em suma, para o desenvolvimento do algoritmo pretendido, iremos utilizar:

- Python, versão 3.5;
- Numpy, versão 1.12.1+mk1;
- Scikit-learn, versão 0.18.2;
- Scipy, versão 0.19.0;
- Tensorflow, versão 1.0.1;
- Pycharm, versão 4.0.5;
- Windows 10 Pro, 64 bits.

A execução da aplicação elaborada no âmbito desta dissertação, exige um elevado poder computacional e tendo em conta as características do portátil em utilização para a execução da aplicação, foi necessário a utilização de uma máquina mais dotada, de forma a que fosse possível a execução dos testes para os *datasets*, que contém maior numero de ficheiros, assim sendo foram utilizadas as seguintes tecnologias:

- Python, versão 3.5.2;
- Numpy, versão 1.13.1+mk1;
- Scikit-learn, versão 0.19.0;
- Scipy, versão 0.19.1;
- Tensorflow, versão 1.3.0;
- Pycharm, versão 2017;
- Windows 10 Pro, 64 bits.

5 Solução

Neste capítulo será apresentada a solução de forma detalhada. Para isso, dividimo-lo nos seguintes subcapítulos: *Dataset*, onde se apresenta a sua composição, nomeadmente a estrutura dos ficheiros existentes; o *Workflow*, onde se descrevem os passos seguidos para a implementação da solução e por fim a Implementação, onde se faz a descrição técnica da solução implementada.

5.1 *Dataset*

Neste projeto, iremos utilizar o *dataset* disponibilizado pelo concurso (Kaggle AM 2016) lançado pela Associação Americana de Epilepsia (Kaggle AM 2016).

Este *dataset*, contém *clipes* de dados de EEG intracranianos de humanos e animais, correspondentes a diferentes estados convulsivos, o estado *preictal* e *interictal*. Para o desenvolvimento da solução no âmbito desta tese de mestrado, apenas são considerados os dados iEEG dos pacientes humanos (Paciente1 e Paciente2). Os dados correspondentes aos animais (Cão1, Cão2, Cão3, Cão4 e Cão5) foram apenas considerados num pequeno número de experiências (Kaggle AM 2016).

Os iEEG's correspondentes ao estado *preictal*, têm uma janela temporal anterior à crise de 1 hora e tem diferentes durações, 5 e 10 minutos. Os iEEG's correspondentes ao estado *interictal* têm uma duração de 4 horas pós e pré-crise convulsiva (Kaggle AM 2016).

As figuras seguintes apresentam todo o processo necessário para a recolha dos sinais iEEG (em pacientes humanos e em animais) que compõem o *dataset.*

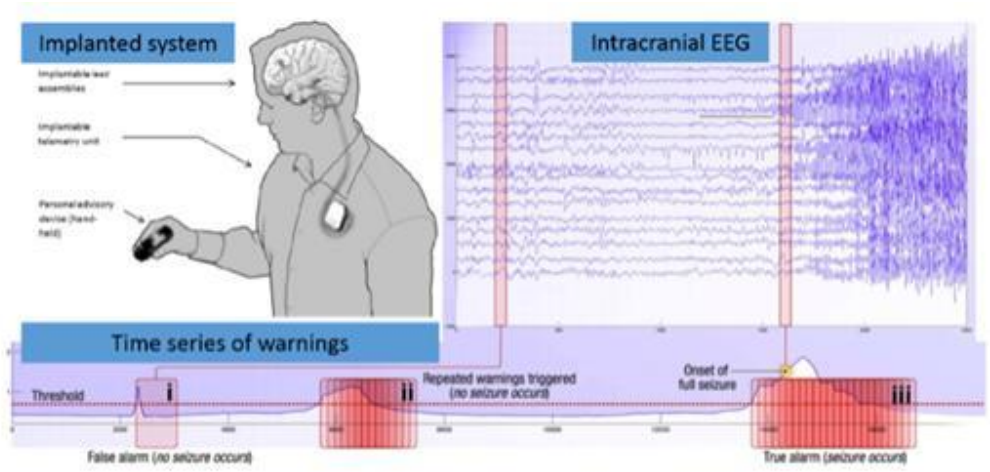

Figura 11 - Exemplo do sistema utilizado nos humanos (Kaggle 2016)

A Figura 11, apresenta a forma como foi realizada a recolha da informação de iEEG nos humanos.

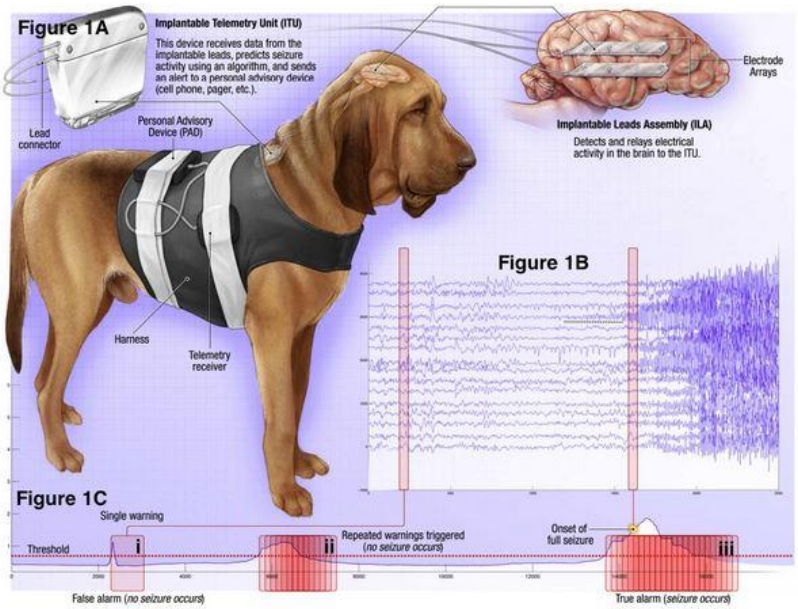

Figura 12 - Exemplo do sistema utilizado nos animais (Kaggle AM 2016)

A Figura 12, apresenta a forma como foi realizada a recolha da informação iEEG nos animais.

Ambas as figuras, demonstram a recolha da informação dos iEEG que iremos utilizar neste trabalho. Estes sistemas recorreram à implantação de dispositivos de telemetria e de recolha de informação. O sistema implantado nos animais, consistiu na utilização de 16 elétrodos, com uma frequência de 400Hz. Nos pacientes, a frequência utilizada foi de 5000Hz (Kaggle AM 2016).

Os dados estão armazenados em ficheiros .mat. Cada ficheiro, contém a seguinte estrutura de dados (Kaggle AM 2016):

- Dados: uma matriz de valores;
- Data_length_sec: duração de cada linha;
- Sampling_frequency: número de amostras correspondentes a 1 segundo de dados EEG;
- Canais: lista de nomes de elétrodos correspondentes às linhas no campo de dados;
- Sequência: índice para cada segmento numa hora.

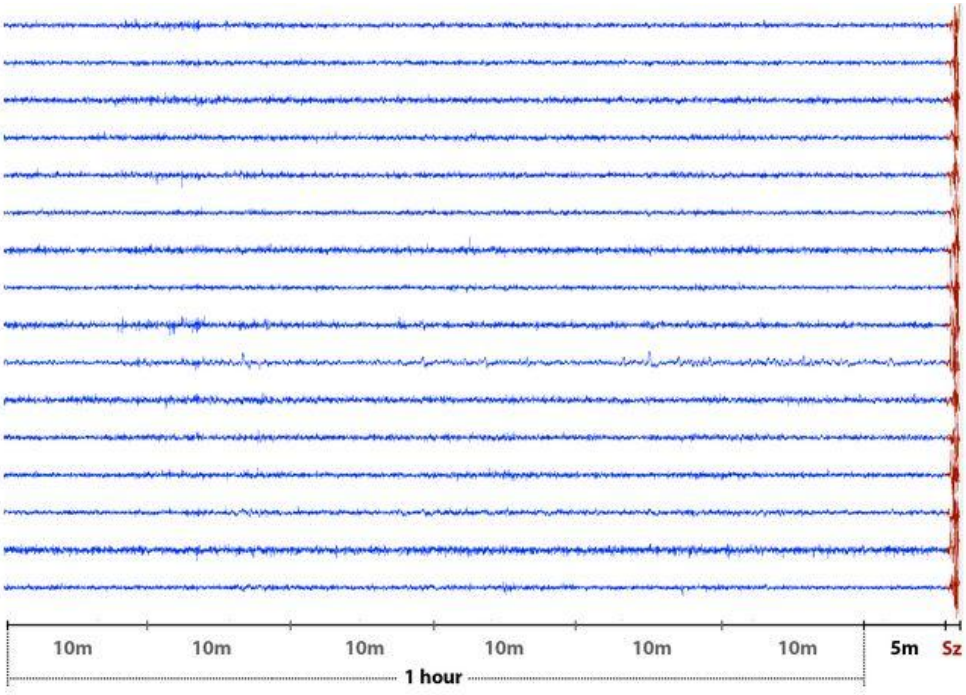

Figura 13 - Amostra do sinal captado por cada elétrodo (Kaggle AM 2016)

A Figura 13, apresenta uma amostra EEG do *dataset* disponibilizado (Kaggle AM 2016). Foram fornecidos partes de 10 minutos de sinais EEG de uma hora, de uma forma aleatória. Nesta imagem, pode observar-se o sinal obtido nos diferentes canais, correspondendo aos elétrodos no cérebro.

5.2 *Workflow*

Este subcapítulo permite descrever o processo realizado no tratamento e utilização dos dados obtidos pelos iEEG, assim como, indicar os passos seguidos até à obtenção final dos valores pretendidos.

Na Figura 14, encontram-se representadas duas possíveis abordagens para análise e classificação de sinais EEG. Conforme indicado no ponto 1.5, será seguida a metodologia *deep learning*, com *Convolutional Neural Networks*, a metodologia b), e será utilizado o Tensorflow para as criar.

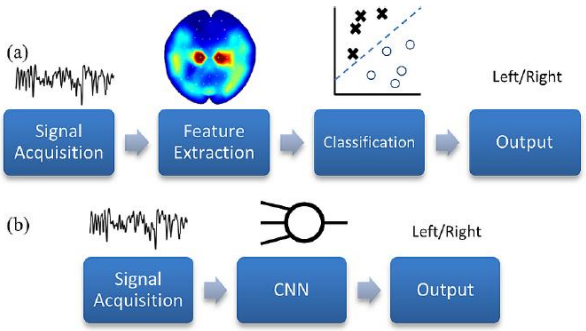

Figura 14 - Metodologias (Nurse et al 2016)

Ambas as abordagens requerem um tratamento dos sinais, tratamento esse designado de pré-processamento.

Na abordagem a), é apresentada a metodologia tradicional (inicialmente pensada para esta dissertação), que consiste nas seguintes fases:

- *Signal Acquisition*: o sinal é adquirido e sujeito a pré-processamento;
- *Feature extraction*: extração de atributos para a análise dos sinais;
- *Classification*: usar métodos de classificação dos sinais obtidos (secção 2.2.3);
- *Output*: obtenção dos valores finais.

Na abordagem b), é apresentada a metodologia que iremos seguir e diferencia-se, pela utilização de *Convolutional Neural Networks* (CNN), para análise e classificação dos sinais EEG. Estudos efetuados mostram que os métodos *deep learning,* têm a capacidade de deteção automática de atributos a partir de uma vasta gama dados. Deste modo, esta abordagem torna-se mais poderosa que uma abordagem focada num número mais reduzido de atributos (Nurse et al 2016).

Cada iEEG, contém séries temporais correspondendo às medições nos diferentes locais espaciais no cérebro. Pretende-se detetar os atributos interessantes que possam existir, estudando esses sinais (Brinkmann et al 2016).

Segundo LeCun, as CNN são do tipo *feedforward* (LeCun et al 1998). A sua forma consiste na passagem de um valor introduzido através de uma ou mais camadas (*layers*) de neurónios ou nós, sendo que cada um desses neurónios representa uma combinação linear desse valor. As combinações lineares, resultam numa camada de saída, através da passagem entre camadas de uma função de ativação tipicamente não linear (Walker 2015).

Existem diversos tipos de camadas que compõem uma CNN. Assim sendo, apresentamos abaixo as camadas mais conhecidas/utilizadas (CS231n CNN 2017).

- *Convolutional,* camada onde é realizada toda a computação necessária e que define os filtros a utilizar;
- *Pooling,* consiste na camada utilizada para redução de escalas, e subdivide-se em *average-pooling* e *max-pooling;*
- *Fully-Connected,* camada onde todos os neurónios têm possibilidade de ter total ligação com as camadas anteriores.

Às imagens, são aplicados filtros na camada *Convolutional* e são subamostradas na camada de *Pooling.* Através do algoritmo *back-propagation* (Rumelhart etal 1988), são calculados os pesos da rede, na camada *Convolutional,* de forma a reduzir os possíveis erros de classificação (LeCun et al 1998), (Tabar et al 2016). O algoritmo *back-propagation* é um algoritmo de treino muito conhecido para redes multicamadas (Gama et al 2012).

Assim, a abordagem que propomos para este trabalho, é composta pelos seguintes passos:

- Conversão dos ficheiros que compõem o *dataset,* correspondentes a cada uma das classes (*preictal* e *interictal*), disponibilizados em formato .mat para formato .bmp, através de um programa em Python;
- As imagens obtidas no passo anterior são utilizadas para definir a arquitetura e os parâmetros da CNN, utilizando o TensorFlow e Python;
- Realização de testes na rede, com as imagens obtidas no passo anterior, e aplicação de uma validação cruzada de 10 *folds* ao *dataset* de imagens;
- Avaliação dos resultados obtidos.

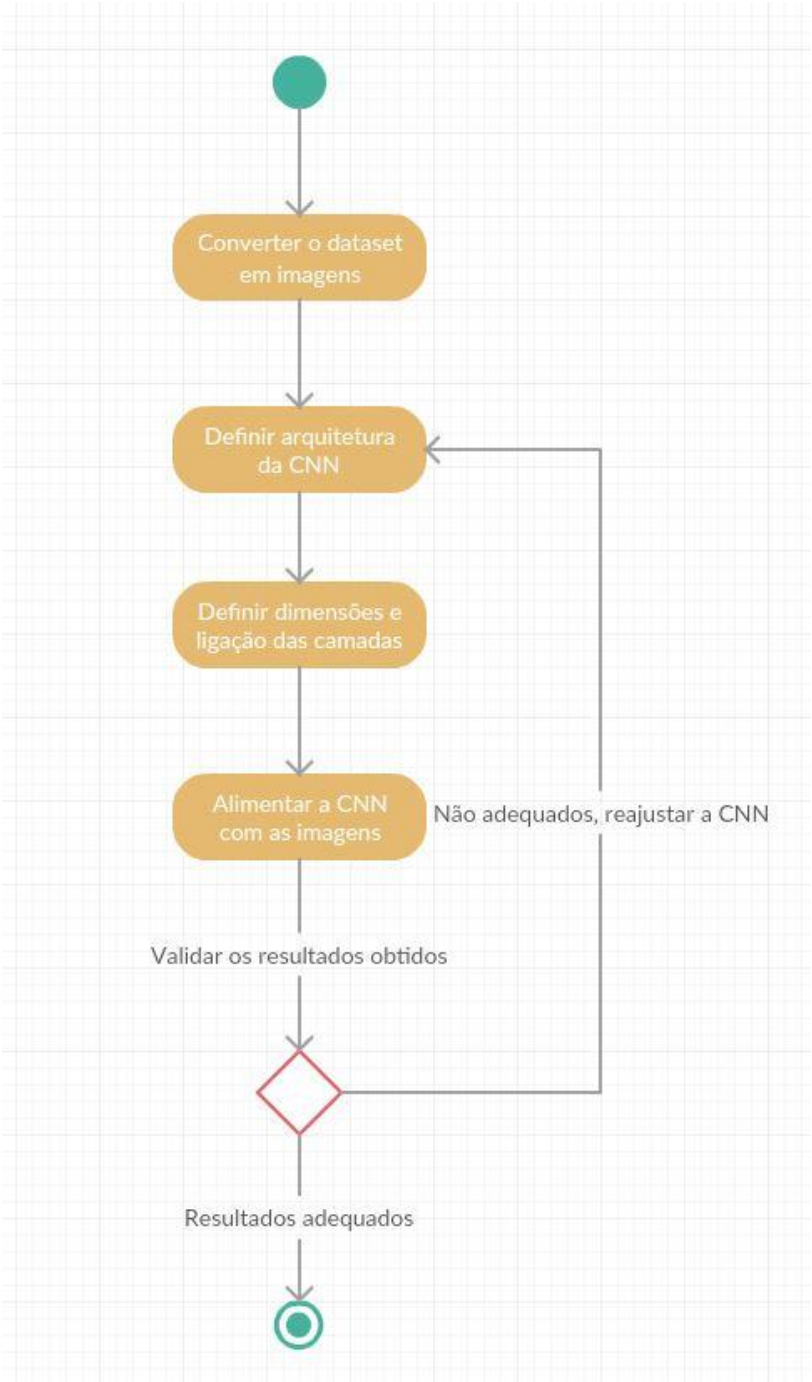

Figura 15 - Diagrama de atividades

Na Figura 15, apresenta-se um diagrama de atividades correspondente aos pontos acima apresentados que permite demonstrar os processos utilizados no desenvolvimento da solução pretendida para esta dissertação.

5.3 Implementação

Neste subcapítulo descreve-se a forma como foi desenvolvida a solução apresentada nesta dissertação, desde a conversão dos *datasets* à explicação da implementação da rede neuronal.

5.3.1 Conversão dos *datasets*

O *dataset*, estava dividido em duas categorias de dados (Kaggle 2016):

- *preictal*: sinais com crise convulsiva
- *interictal*: sinais sem crise convulsiva

Na Tabela 4, podemos observar a quantidade dos exemplos acima descritos, presentes em cada um dos conjunto de dados, assim como o número de canais utilizados na gravação dos sinais intracranianos.

Tabela 4 - Composição dos Datasets

Dataset	***Interictal***	***Preictal***	**Número de canais**
Paciente 1	50	18	15
Paciente 2	42	18	24

A solução implementada para a conversão dos *datasets*, consistiu na implementação de um excerto de código Python, recorrendo a métodos disponibilizados pelos módulos scikit-learn[4] e scipy[5]. Este excerto de código, converteu todos os ficheiros .mat existentes para cada um dos casos de estudo, em ficheiros imagem do tipo .bmp, essa solução foi implementada de forma genérica, permitindo assim que independentemente do *dataset*, as imagens seriam sempre geradas, pois os ficheiros dos diferentes *datasets*, não tem o mesmo tamanho da matriz de informação.

5.3.2 Implementação da rede neuronal

Este subcapítulo permite descrever, a arquitetura da rede, através dos seguintes pontos: ligações das camadas, número de camadas e a composição de cada camada e o treino realizado sobre os *datasets.* A rede neuronal foi implementada recorrendo a uma CNN, para tratar os dados iEEG, convertidos em imagens.

[4] http://scikit-learn.org/stable/index.html
[5] https://www.scipy.org/

5.3.2.1 Arquitetura

Numa primeira fase, a arquitetura definida para a nossa solução foi composta por 5 camadas convulsionais, com a Relu como função de ativação e o Adam optimizer como função de treino, e 4 camadas de *max-pooling*. As 4 camadas de *pooling*, precedem as primeiras 4 camadas convulsionais, sendo a 5º e última, uma camada *fully connected*, seguida do *dropout* e sobre o qual é aplicada o algoritmo *softmax*.

As camadas convulsionais, utilizam campos 5x5 com um *stride* de 1 pixel e não tem qualquer adição de pixéis, para garantir que ambas as imagens têm o mesmo tamanho. As camadas de *max-pooling*, são feitas sobre uma janela de 2x2, com um *stride* de 2 pixéis.

Na Figura 16, podemos observar a arquitetura descrita.

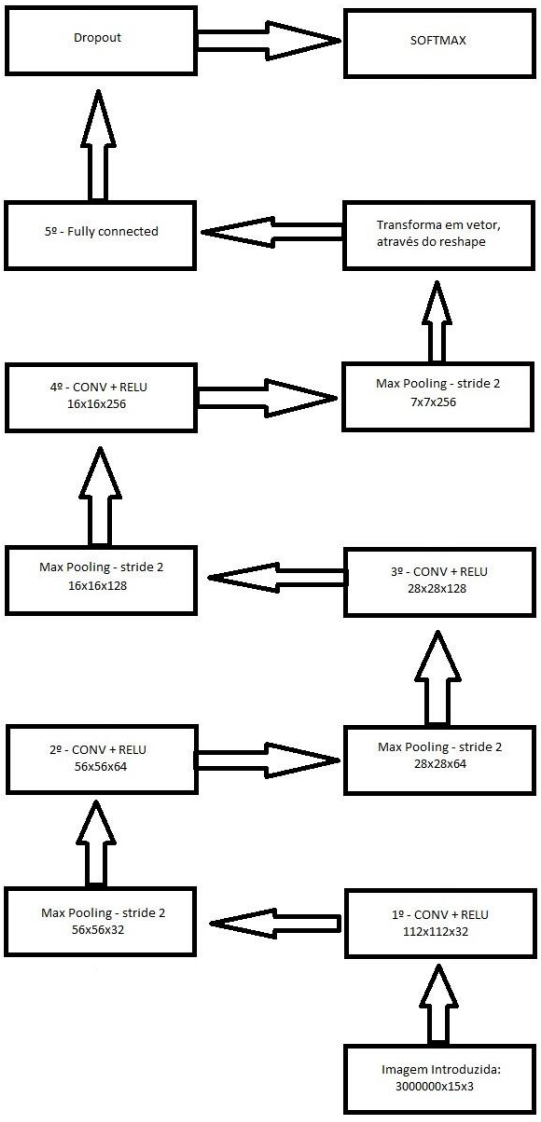

Figura 16 - Arquitetura Inicial

5.3.2.2 Treino

O treino da rede foi feito com recurso ao método de validação cruzada de 10 *folds*, designado de StratifiedKFold[6] com o parâmetro *shuffle* a *true*, para garantir que os dados seriam misturados antes da separação, e o *random_seed* a um valor inteiro, para garantir que todos os testes seriam sempre realizados sobre os mesmos elementos.

Após os primeiros testes realizados, conseguimos compreender que os resultados obtidos necessitavam de ser melhorados, dessa forma foram utilizadas outras função de ativação, assim como outra função de Treino. As funções de ativação utilizadas foram a Tangente Hiperbólica, Sigmoid, SoftPlus e a função de treino, o Gradient descent. Estas alterações na arquitetura da rede, permitiram que conseguíssemos obter outros resultados para comparação e após a obtenção desses resultados estudar qual ou quais os melhores. No capitulo 6.2, podemos observar os diferentes resultados obtidos por cada uma das funções ativação e de treino utilizadas.

[6] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

6 Avaliação

Neste capítulo, iremos descrever como avaliar os resultados obtidos, quais as medidas e metodologias de avaliação a utilizar.

6.1 Medidas de avaliação, Hipóteses e metodologias de avaliação

As medidas a utilizar na avaliação da solução serão a taxa de acerto (*accuracy*), *F-Measure* e *Area Under the ROC Curve* (AUC), para comparar os modelos de classificação. Poderão ainda ser utilizadas as medidas Sensibilidade e Especificidade, visto surgirem com frequência na bibliografia. Estas medidas estão explicadas com detalhe no subtópico 2.2.4.

No sentido de decidir qual o melhor modelo de classificação serão comparadas várias parametrizações de configuração de *Convolutional Neural Networks* para as diferentes medidas de avaliação.

Nas experiências efetuadas foi utilizada a metodologia de validação cruzada com a utilização de 10 subconjuntos, e será testada segundo as medidas de avaliação referidas na secção anterior.

A hipótese nula, afirma que os desempenhos dos modelos são equivalentes. A hipótese alternativa é que pelo menos um é diferente (o melhor). Ou seja, teremos a igualdade das médias das medidas de avaliação utilizadas (ou a mediana, dependendo do tipo de teste a realizar) contra a hipótese alternativa de que o modelo apresentado é melhor.

6.2 Avaliação

Neste subcapítulo, iremos apresentar as tabelas com os resultados obtidos pelas CNN nas experiências efetuadas para cada um dos *datasets* utilizados, Paciente 1 e 2. As tabelas serão divididas em 7 colunas, colunas essas que tem o seguinte significado:

- Função de ativação*,* Contém as diferentes funções de ativação utilizadas;
- Função de treino*,* Contém as diferentes funções de treino utilizadas;
- *Accuracy,* representa a taxa de acerto do modelo nos exemplos de teste*;*
- AUC*,* representa o valor da área sob a curva ROC obtido*;*
- *Recall,* obtido pelo modelo (taxa de acerto para a classe positiva)*;*
- *Precision,* obtida pelo modelo (classificar uma amostra positiva como sendo negativa)*;*
- *F-Measure,* obtida pelo modelo*.*

Os testes aos *datasets* dos Pacientes serão realizados sobre imagens não separadas por canais, completas, e separadas por canais. A cada caso de teste será aplicado um *shuffle*, que indica se antes da separação das imagens em conjunto de treino/conjunto de teste, estas foram ordenadas aleatoriamente ou não.

Os testes aos *datasets* dos cães serão realizados apenas sobre as imagens não separadas por canais, uma vez que dada a enorme descrepância existente entre os ficheiros de cada classe, os resultados obtidos não são nada favoráveis para as conclusões que pretendemos retirar e estes mesmos resultados são praticamente iguais aos já obtidos para as imagens completas.

Na tabelas abaixo apresentadas, apresentam-se os resultados para as imagens completas com *shuffle* e imagens separadas por canais, também com *shuffle*. Como se tratam de *datasets* com exemplos desbalanceados, para que o resultado possa ser considerado interessante, a taxa de acerto preditiva do classificador deve ser superior à taxa de acerto correspondente a atribuir a classe maioritária a todos os exemplos.

6.2.1 Paciente 1

Neste *dataset* existem 50 exemplos da classe *interictal* e 18 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.735.

Tabela 5 - Resultados Paciente 1 para as imagens completas com *shuffle*

Função ativação	**Função treino**	***Accuracy***	**AUC**	***Recall***	***Precision***	***F-Measure***
Relu	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0

Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
Softplus	Adam optimizer	0.664	0.59	0.78/0.4	0.728/0.193	0.714/0.241
Tanh	Adam optimizer	0.869	0.79	0.98/0.6	0.880/0.65	0.920/0.6
Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.04
Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.08
Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
Tanh	Gradient descent	0.542	0.5	0.6/0.4	0.440/0.102	0.507/0.161

Como pode ser observado na Tabela 5, os melhores resultados para o Paciente 1, foram obtidos usando a função de ativação Tanh e a função de treino Adam optimizer. Para esta combinação obteve-se uma A*ccuracy* de 0.869, uma AUC igual a 0.79, o valor de *Recall* para a classe negativa de 0.98 e de 0.6 para a classe positiva, o valor de precisão para a classe negativa 0.880 e de 0.65 para a classe positiva e um valor de *F-Measure* para a classe negativa de 0.920 e de 0.6 para a classe positiva.

Na Tabela 6, apresentam-se os resultados obtidos para a mesma combinação de funções, utilizando a medida de avaliação AUC. No Anexo 1, é possivel observar os resultados completos obtidos nos testes.

Tabela 6 - Resultados da AUC do Paciente 1 nas imagens separadas por canais

	Adam optimizer				**Gradient descent**			
Canal	**Relu**	**Sigmoid**	**Softplus**	**Tanh**	**Relu**	**Sigmoid**	**Softplus**	**Tanh**
1	0,59	0,5	0,615	0,875	0,5	0,5	0,5	0,51
2	0,655	0,495	0,61	0,875	0,5	0,5	0,5	0,5
3	0,585	0,5	0,605	0,74	0,5	0,5	0,5	0,46
4	0,635	0,5	0,605	0,875	0,5	0,5	0,5	0,5
5	0,57	0,5	0,67	0,85	0,5	0,5	0,5	0,5
6	0,565	0,5	0,565	0,89	0,5	0,5	0,5	0,5
7	0,475	0,5	0,55	0,85	0,5	0,5	0,5	0,5
8	0,58	0,5	0,685	0,815	0,5	0,5	0,5	0,5
9	0,54	0,5	0,565	0,765	0,5	0,5	0,5	0,5
10	0,55	0,5	0,535	0,875	0,5	0,5	0,5	0,5
11	0,605	0,5	0,575	0,8	0,5	0,5	0,5	0,5
12	0,605	0,5	0,56	0,85	0,5	0,5	0,5	0,5
13	0,62	0,5	0,5	0,815	0,5	0,5	0,5	0,5
14	0,625	0,5	0,57	0,865	0,5	0,5	0,5	0,5
15	0,56	0,5	0,595	0,85	0,5	0,5	0,5	0,5

Na Tabela 6, é possível analisar os resultados obtidos combinando as funções de ativação Relu, Sigmoid, Softplus e Tangente hiperbólica (TanH) e as funções de treino Adam optimizer e Gradient descent. O melhor resultado é sempre obtido para a combinação Tangente hiperbólica e Adam optimizer (0,89 de AUC).

Os resultados obtidos utilizando Adam optimizer e Softplus, apesar de mais fracos, conseguem ser melhores dos que as restantes combinações (0,685 de AUC).Ainda relativamente à função de treino Adam optimizer, observa-se que os piores resultados são obtidos para a função Sigmoid. Para a função de treino Gradient descent, os resultados são bastante fracos.

Na Tabela 7, apresentam-se os resultados obtidos para a mesma combinação de funções, utilizando a medida de avaliação *F-Measure*. Tal como para a AUC, o melhor resultado é sempre obtido para a combinação Tangente hiperbólica e Adam optimizer (0,8).

Tabela 7 - Resultados da *F-Measure* do Paciente 1 nas imagens separadas por canais

	Adam optimizer				**Gradient descent**			
Canal	**Relu**	**Sigmoid**	**Softplus**	**Tanh**	**Relu**	**Sigmoid**	**Softplus**	**Tanh**
1	0,216	0	0,233	0,8	0,044	0,088	0	0,138
2	0,316	0,033	0,237	0,766	0,044	0,133	0	0,117
3	0,217	0,044	0,277	0,5	0	0,088	0,044	0,117
4	0,323	0	0,266	0,766	0	0,088	0,044	0,133
5	0,153	0	0,383	0,733	0,044	0,117	0,044	0,117
6	0,14	0	0,166	0,8	0,044	0,117	0,044	0,117
7	0,073	0	0,178	0,733	0,044	0,088	0	0,088
8	0,219	0	0,403	0,646	0	0,117	0	0,133
9	0,123	0,044	0,183	0,546	0	0,088	0,044	0,133
10	0,135	0	0,106	0,766	0,044	0,117	0,044	0,117
11	0,24	0	0,2	0,6	0	0,088	0	0,133
12	0,257	0	0,18	0,733	0	0,088	0	0,161
13	0,3	0	0,044	0,666	0	0,088	0	0,117
14	0,323	0	0,233	0,783	0	0,117	0,044	0,133
15	0,194	0	0,213	0,7	0	0,133	0	0,117

Os resultados obtidos utilizando Adam optimizer e Softplus, apesar de mais fracos, conseguem ser melhores dos que as restantes combinações (0,403 de *F-Measure*). Ainda relativamente à função de treino Adam optimizer, observa-se que os piores resultados são obtidos para a função Sigmoid. Para a função de treino Gradient descent, os resultados são bastante fracos.

Na Tabela 8, apresentam-se os dez melhores resultados ordenados por AUC.

Tabela 8 - Resultados Paciente 1 imagens separadas por canais, ordenado por AUC

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	F1
6	Tanh	Adam optimizer	0.926	0.89	0.9/0.8	0.938/0.85	0.8
1	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.921/0.9	0.8
2	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.766
4	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.766
10	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.766
14	Tanh	Adam optimizer	0.914	0.865	0.98/0.75	0.918/0.85	0.783
5	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.909/0.8	0.733
7	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.909/0.8	0.733
12	Tanh	Adam optimizer	0.915	0.85	1.0/0.7	0.905/0.8	0.733
15	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.914/0.7	0.7

Pode observar-se, novamente, que os melhores resultados foram obtidos com a função de ativação Tangente hiperbólica e com a função de treino Adam optimizer. O melhor resultado para a medida de avaliação AUC foi 0.89.

Na Tabela 9, apresentam-se os dez melhores resultados ordenados por *F-Measure* da classe positiva.

Tabela 9 - Resultados Paciente 1 imagens separadas por canais, ordenado por *F-Measure*

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	F1
6	Tanh	Adam optimizer	0,926	0,89	0,98/0,8	0,938/0,85	0,8
1	Tanh	Adam optimizer	0,928	0,875	1,0/0,75	0,921/0,9	0,8
14	Tanh	Adam optimizer	0,914	0,865	0,98/0,75	0,918/0,85	0,783
2	Tanh	Adam optimizer	0,928	0,875	1,0/0,75	0,926/0,8	0,766
4	Tanh	Adam optimizer	0,928	0,875	1,0/0,75	0,926/0,8	0,766
10	Tanh	Adam	0,928	0,875	1,0/0,75	0,926/0,8	0,766

		optimizer					
5	Tanh	Adam optimizer	0,914	0,85	1,0/0,7	0,909/0,8	0,733
7	Tanh	Adam optimizer	0,914	0,85	1,0/0,7	0,909/0,8	0,733
12	Tanh	Adam optimizer	0,914	0,85	1,0/0,7	0,905/0,8	0,733
15	Tanh	Adam optimizer	0,914	0,85	1,0/0,7	0,914/0,7	0,7

Tal como os resultados obtidos para a medida de avaliação AUC, os melhores resultados para a medida *F-Measure* (0,8) foram, novamente, obtidos com a função de ativação Tangente hiperbólica e com a função de treino Adam optimizer.

Comparando os resultados obtidos por canal com os resultados obtidos quando considerados os canais em conjunto, observa-se que se obtêm melhores resultados na situação de canais separados (0,89 de AUC e 0,8 de *F-Measure*) do que com os canais todos juntos (0,79 de AUC e 0,8 de *F-Measure*).

Quando comparados com os resultados obtidos anteriormente para o mesmo problema (Martins 2016), não usando técnicas de *deep learning*, observa-se que os resultados obtidos neste trabalho, para o Paciente 1, são melhores. No trabalho anterior, a melhor AUC foi 0,847 obtida usando o algoritmo de classificação *Random Forest*.

6.2.2 Paciente 2

No *dataset* Paciente 2, existem 42 exemplos da classe *interictal* e 18 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.7.

Tabela 10 - Resultados Paciente 2 para as imagens completas com *shuffle*

Função Ativação	**Função treino**	***Accuracy***	**AUC**	***Recall***	***Precision***	***F-Measure***
Relu	Adam optimizer	0.702	0.575	0.85/0.3	0.791/0.24	0.775/0.240
Sigmoid	Adam optimizer	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
Softplus	Adam optimizer	0.719	0.55	0.95/0.15	0.731/0.25	0.823/0.183
Tanh	Adam optimizer	0.809	0.75	0.9/0.6	0.869/0.6	0.857/0.566
Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
Softplus	Gradient	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044

Função Ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
	descent					
Tanh	Gradient descent	0.561	0.505	0.66/0.35	0.493/0.12	0.558/0.173

Como pode ser observado na Tabela 10, os melhores resultados para o Paciente 2, foram obtidos usando a função de ativação Tanh e a função de treino Adam optimizer. Para esta combinação obteve-se uma *Accuracy* de 0.809, uma AUC igual a 0.75, o valor de *Recall* para a classe negativa de 0.9 e de 0.6 para a classe positiva, o valor de precisão para a classe negativa 0.869 e de 0.6 para a classe positiva e um valor de *F-Measure* para a classe negativa de 0.857 e de 0.566 para a classe positiva.

Na Tabela 11, apresentam-se os resultados obtidos para a mesma combinação de funções, utilizando a medida de avaliação AUC. No Anexo 2, é possivel observar os resultados completos obtidos nos testes.

Tabela 11 - Resultados AUC do Paciente2 nas imagens separadas por canais

	Adam optimizer				**Gradient descent**			
Canal	**Relu**	**Sigmoid**	**Softplus**	**Tanh**	**Relu**	**Sigmoid**	**Softplus**	**Tanh**
1	0,575	0,5	0,632	0,887	0,5	0,5	0,525	0,5
2	0,637	0,49	0,537	0,865	0,5	0,5	0,5	0,5
3	0,575	0,5	0,525	0,737	0,5	0,5	0,5	0,5
4	0,667	0,5	0,587	0,9	0,5	0,5	0,5	0,5
5	0,625	0,5	0,712	0,837	0,5	0,5	0,5	0,49
6	0,577	0,55	0,575	0,875	0,5	0,5	0,5	0,5
7	0,537	0,5	0,687	0,9	0,5	0,5	0,54	0,5
8	0,65	0,5	0,6	0,8	0,5	0,5	0,5	0,5
9	0,5	0,525	0,537	0,875	0,5	0,5	0,5	0,487
10	0,567	0,5	0,525	0,892	0,5	0,5	0,5	0,5
11	0,562	0,5	0,575	0,862	0,5	0,5	0,5	0,5
12	0,525	0,5	0,575	0,885	0,5	0,5	0,5	0,5
13	0,562	0,5	0,547	0,84	0,5	0,5	0,5	0,51
14	0,587	0,5	0,525	0,912	0,5	0,5	0,5	0,5
15	0,6	0,562	0,6	0,85	0,5	0,5	0,5	0,5
16	0,64	0,49	0,537	0,845	0,5	0,5	0,5	0,5
17	0,662	0,5	0,66	0,805	0,5	0,5	0,5	0,5
18	0,675	0,5	0,65	0,877	0,5	0,5	0,5	0,5
19	0,562	0,5	0,575	0,802	0,5	0,5	0,5	0,5
20	0,55	0,55	0,562	0,9	0,5	0,5	0,5	0,5
21	0,762	0,512	0,537	0,865	0,5	0,5	0,5	0,5
22	0,562	0,5	0,6	0,875	0,5	0,5	0,5	0,5
23	0,562	0,5	0,625	0,812	0,5	0,5	0,5	0,5
24	0,65	0,5	0,662	0,8	0,5	0,5	0,5	0,5

Na Tabela 11, tal como para o Paciente 1, é possível analisar os resultados obtidos combinando as funções de ativação Relu, Sigmoid, Softplus e Tangente hiperbólica (TanH) e as funções de treino Adam optimizer e Gradient descent. O melhor resultado é sempre obtido para a combinação Tangente hiperbólica e Adam optimizer (0,912 de AUC).

Os resultados obtidos utilizando Adam optimizer com Softplus e Relu, apesar de mais fracos, conseguem ser melhores dos que as restantes combinações (0,712 de AUC usando Softplus e 0,762 para Relu). Ainda relativamente à função de treino Adam optimizer, observa-se que os piores resultados são obtidos para a função Sigmoid. Para a função de treino Gradient descent, os resultados são bastante fracos, tal como no *dataset* Paciente 1.

Na Tabela 12, apresentam-se os resultados obtidos para a mesma combinação de funções, utilizando a medida de avaliação *F-Measure*. Tal como para a AUC, o melhor resultado é sempre obtido para a combinação Tangente Hiperbólica e Adam optimizer (0,846).

Tabela 12 - Resultados *F-Measure* do Paciente2 nas imagens separadas por canais

	Adam optimizer				**Gradient descent**			
Canal	Relu	Sigmoid	Softplus	Tanh	Relu	Sigmoid	Softplus	Tanh
1	0,253	0	0,333	0,78	0	0,133	0,066	0,144
2	0,344	0,044	0,163	0,8	0	0,144	0,044	0,177
3	0,196	0	0,111	0,5	0	0,127	0,044	0,133
4	0,346	0	0,25	0,8	0	0,144	0	0,177
5	0,3	0	0,433	0,713	0	0,144	0	0,177
6	0,216	0,1	0,206	0,766	0,044	0,177	0	0,144
7	0,19	0	0,393	0,8	0	0,144	0,08	0,144
8	0,366	0	0,226	0,6	0	0,133	0	0,177
9	0,066	0,066	0,137	0,766	0,044	0,127	0,044	0,177
10	0,233	0,1	0,183	0,82	0	0,127	0	0,127
11	0,18	0,044	0,25	0,746	0	0,133	0	0,194
12	0,14	0	0,211	0,846	0	0,144	0	0,222
13	0,196	0,05	0,178	0,733	0,044	0,133	0	0,183
14	0,263	0	0,163	0,846	0,044	0,144	0	0,127
15	0,313	0,146	0,322	0,7	0	0,127	0,044	0,144
16	0,373	0	0,207	0,766	0,044	0,133	0,044	0,177
17	0,438	0,044	0,475	0,666	0,044	0,127	0,044	0,177
18	0,437	0,044	0,35	0,813	0	0,127	0,044	0,133
19	0,163	0	0,18	0,646	0,044	0,133	0,044	0,177
20	0,133	0,137	0,196	0,8	0	0,144	0,044	0,144
21	0,583	0,107	0,173	0,76	0	0,133	0	0,177
22	0,19	0	0,29	0,793	0	0,133	0	0,177
23	0,221	0	0,337	0,646	0	0,133	0,044	0,177
24	0,363	0	0,366	0,633	0,044	0,127	0	0,262

Os resultados obtidos utilizando Adam optimizer com Softplus e Relu, apesar de mais fracos, conseguem ser melhores dos que as restantes combinações (0,583 de *F-Measure* usando Relu e 0,475 para Softplus). Ainda relativamente à função de treino Adam optimizer, observa-se que os piores resultados são obtidos para a função Sigmoid. Para a função de treino Gradient descent, os resultados são bastante fracos, tal como no *dataset* Paciente 1.

Na Tabela 13, apresentam-se os dez melhores resultados ordenados por AUC.

Tabela 13 - Resultados Paciente2 imagens separadas por canais, ordenado por AUC

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	F1
14	Tanh	Adam optimizer	0,94	0,912	0,975/0,85	0,954/0,866	0,846
4	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
7	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
20	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
10	Tanh	Adam optimizer	0,911	0,892	0,935/0,85	0,946/0,8	0,82
1	Tanh	Adam optimizer	0,926	0,887	0,975/0,8	0,942/0,766	0,78
12	Tanh	Adam optimizer	0,897	0,885	0,92/0,85	0,93/0,866	0,846
18	Tanh	Adam optimizer	0,907	0,877	0,955/0,8	0,926/0,866	0,813
6	Tanh	Adam optimizer	0,926	0,875	1,0/0,75	0,922/0,8	0,766
9	Tanh	Adam optimizer	0,926	0,875	1,0/0,75	0,922/0,8	0,766

Analisando a tabela, observa-se que os melhores resultados foram, novamente, obtidos com a função de ativação Tangente hiperbólica e com a função de treino Adam optimizer. O melhor resultado para a medida de avaliação AUC foi 0.912.

Na Tabela 14, apresentam-se os dez melhores resultados ordenados por *F-Measure* para a classe positiva.

Tabela 14 - Resultados Paciente2 imagens separadas por canais, ordenado por *F-Measure*

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	F1
14	Tanh	Adam optimizer	0,94	0,912	0,975/0,85	0,954/0,866	0,846
12	Tanh	Adam optimizer	0,897	0,885	0,92/0,85	0,93/0,866	0,846
10	Tanh	Adam optimizer	0,911	0,892	0,935/0,85	0,946/0,8	0,82
18	Tanh	Adam optimizer	0,907	0,877	0,955/0,8	0,926/0,866	0,813
4	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
7	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
20	Tanh	Adam optimizer	0,942	0,9	1,0/0,8	0,942/0,8	0,8
1	Tanh	Adam optimizer	0,926	0,887	0,975/0,8	0,942/0,766	0,78
6	Tanh	Adam optimizer	0,926	0,875	1,0/0,75	0,922/0,8	0,766
9	Tanh	Adam optimizer	0,926	0,875	1,0/0,75	0,922/0,8	0,766

Os melhores resultados para a medida *F-Measure* (0,846) foram obtidos, novamente, com a função de ativação Tangente hiperbólica e com a função de treino Adam optimizer.

Comparando os resultados obtidos por canal com os resultados obtidos quando considerados os canais em conjunto, observa-se que se obtêm melhores resultados na situação de canais separados (0,912 de AUC e 0,846 de *F-Measure*) do que com os canais todos juntos (0,75 de AUC e 0,6 de *F-Measure)*.

Tal como no caso do conjunto de dados Paciente 1, comparando os resultados obtidos com os resultados de um trabalho anterior (Martins 2016), não usando técnicas de *deep learning*, observa-se que os resultados obtidos neste trabalho, para o Paciente 2, também são melhores. No trabalho anterior, a melhor *F-Measure* foi 0,604 obtida usando o algoritmo de classificação SVM.

Comparando também com alguns dos resultados, possiveis de observar, nos estudos documentados na Tabela 1, destaca-se os resultados apresentados no estudo (Brinkmann et al 2016).

Tabela 15 - Resultados AUC top 10 finalistas Kaggle.com (Brinkmann et al 2016)

Posição	Nome da Equipa	Leaderboard Publica	Leaderboard Privada
1	QMSDP	0.86	0.82
2	Birchwood	0.84	0.80
3	ESAI CEU-UCH	0.82	0.79
4	Michael Hills	0.86	0.79
5	KPZZ	0.82	0.79
6	Carlos Fernandez	0.84	0.79
7	Isaac	0.84	0.79
8	Wei Wu	0.82	0.79
9	Golondrina	0.82	0.78
10	Sky Kerzner	0.84	0.78

Estes resultados, presentes na Tabela 15, foram obtidos nos datasets de 5 cães e 2 pacientes, pertencentes ao concurso kaggle (já referido), e utilizando as metodologias descritas na Tabela 2. Note-se que estes resultados são apenas indicadores e não podem ser diretamente comparados por não sabermos ao certo, como foram calculados os resultados obtidos.

6.2.3 Outras experiências

Foram realizadas outras experiências com os sinais obtidos dos cães (cinco). Apesar das diferentes tentativas na configuração da rede, todas elas obtiveram valores de AUC muito próximo de 0.5. Estes maus resultados devem-se possivelmente ao grande desbalanceamento das classes positiva e negativa nestes conjuntos de dados (ver tabelas no Anexo 3).

6.2.4 Teste estatístico

Foram apresentados e discutidos os resultados obtidos, sendo que os melhores resultados (de longe) foram obtidos para o modelo que combina a função de ativação Tangente hiperbólica e com a função de treino Adam optimizer. Para suportar a afirmação de que este modelo de classificação é melhor quando comparado com outros modelos, devem ser utilizados testes estatísticos.

Para testar vários modelos em simultâneo, Demsár recomenda o teste de Friedman (Demsar 2006). Assim, neste caso, seria de usar o teste de Friedman (teste não paramétrico) para comparação dos modelos (seguido do teste de Nemenyi, um teste post-hoc).

7 Conclusão

Neste trabalho, foi desenvolvido um protótipo de um modelo possível para um sistema de deteção e previsão de convulsões epiléticas, com o objetivo de garantir previsões eficazes e seguras dos momentos de crise convulsiva e de forma a proporcionar uma melhor qualidade de vida a todos os seus pacientes.

Com a elaboração deste modelo, foi possível observar através dos diferentes resultados obtidos, para os diferentes *datasets*, que quando comparados com outras tecnologias aplicadas a este problema, este sistema apresentou melhores resultados e que poderá vir a dar mais provas em implementações futuras, da metodologia proposta no âmbito do tema desta dissertação de mestrado.

A implementação da solução consistiu na análise de ficheiros iEEG, posteriormente convertidos em imagem do tipo .bmp. Essas imagens, seriam depois "catalogadas" para a utilização da aprendizagem supervisionada, documentada na subcapítulo 2.1 e posteriormente alimentariam a rede CNN, para a obtenção dos resultados da classificação.

Relativamente à avaliação da solução, é possível concluir que nos diferentes testes realizados aos *datasets* existentes, que os melhores resultados foram obtidos utilizando a Tangente hiperbólica como função de ativação combinada com a função de treino Adam optimizer, em todos os casos de estudo. Os resultados obtidos para todos os *datasets* dos pacientes foram mais satisfatórios a nível geral, do que os obtidos para os cães. O desbalanceamento nas classes, *preictal* e *interictal*, era bastante superior no caso dos cães, conduzindo a resultados pouco superiores aos de uma classificação aleatória (AUC muito próxima de 0.5).

7.1 Trabalho futuro

Neste trabalho, foi proposto um modelo para um sistema de deteção e previsão de convulsões epiléticas que poderá ser utilizado para a elaboração de novas soluções neste âmbito. Numa perspetiva de melhoria deste trabalho, são de considerar as seguintes questões:

- Modificar e avaliar algumas das parametrizações utilizadas, nomeadamente o tamanho da matriz de *pooling,* acréscimo de mais camadas convulsionais, maior tamanho das imagens a alimentar a CNN, entre outros;
- Comparação com outras soluções existentes;
- Avaliar técnicas a aplicar a *datasets* bastante desbalanceados, como o *oversampling* ou *undersampling*;
- Melhorias a nível da exigência computacional que este sistema exige, permitindo que fosse possível utilizar em algo portátil, como por exemplo um *smartband* ou um *smartwatch*, ou *smartphone*, ou até mesmo um equipamento portátil especialmente desenhado para o efeito;
- Realizar mais testes com mais *datasets*;
- Garantir monitorizações em tempo real com alertas, no equipamento pretendido, de forma a que o paciente pudesse atuar no momento certo.

Referências

(CUF 2017) CUF, Epilepsia, visto a 14 Janeiro 2017, https://www.saudecuf.pt/mais-saude/doencas-a-z/epilepsia

(Cook et al 2013) Cook MJ, O'Brien TJ, Berkovic SF, Murphy M, Morokoff A, Fabinyi G, D'Souza W, Yerra R, Archer J, Litewka L, Hosking S, Lightfoot P, Ruedebusch V, Sheffield WD, Snyder D, Leyde K, Himes D., 2013. Prediction of seizure likelihood with a long-term, implanted seizure advisory system in patients with drug-resistant epilepsy: a first-in-man study. LANCET NEUROL 12:563-571

(Brinkmann et al 2016) Brinkmann, B. H., Wagenaar, J., Abbot, D., Adkins, P., Bosshard, S. C., Chen, M., ... & Pardo, J. 2016 Crowdsourcing reproducible seizure forecasting in human and canine epilepsy.

(Gadhoumi et al 2015) Gadhoumi, K., Lina, J. M., Mormann, F., & Gotman, J.,2015. Seizure prediction for therapeutic devices: A review.

(Karoly et al 2016) Karoly, P. J., Freestone, D. R., Boston, R., Grayden, D. B., Himes, D., Leyde, K., ... & Cook, M. J, 2016. Interictal spikes and epileptic seizures: their relationship and underlying rhythmicity.

(Andrzejak et al 2009) Andrzejak RG, Chicharro D, Elger CE, Mormann F, 2009.Seizure prediction: Any better than chance?

(Snyder et al 2008) Snyder DE, Echauz J, Grimes DB, Litt B, 2008.The statistics of a practical seizure warning system.

(Mormann et al 2007) Mormann, F., Andrzejak, R. G., Elger, C. E., & Lehnertz, K. (2007). Seizure prediction: the long and winding road. Brain, 130(2), 314-333.

(Haut et al 1999) Haut S, Shinnar S, Moshe SL, O'Dell C, Legatt AD, 1999.The association between seizure clustering and status epilepticus in patients with intractable complex partial seizures.

(Deep Learning 2017) deeplearning, Deep Learning, visto a 15 Janeiro 2017, http://deeplearning.net/

(Python 2017) Python Software Foundation Python, visto a 15 Janeiro 2017, https://www.python.org/

(Tensorflow 2017) Tensorflow, Tensorflow, visto a 15 Janeiro 2017, https://www.tensorflow.org/

(Java 2017) Oracle, Java, visto a 18 Janeiro 2017, https://www.oracle.com/java/technologies/index.html

(Tiobe 2017) Tiobe, Tiobe, visto a 18 Janeiro 2017 http://www.tiobe.com/tiobe-index/

(Aashit et al 2014) Aashit K. Shah, Sandeep Mittal , 2014. Invasive electroencephalography monitoring: Indications and presurgical planning. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4001224/

(Kaggle 2016) Kaggle, Melbourne University AES/MathWorks/NIH Seizure Prediction, visto a 30 Novembro 2016, https://www.kaggle.com/c/melbourne-university-seizure-prediction

(Gomes et al., 2014) Gomes, E.F., Jorge, A.M., Azevedo, P.J., 2014. Classifying heart sounds classification using SAX motifs, random forests and text mining techniques. Presented at the Proceedings of the 18th International Database Engineering & Applications Symposium, ACM, pp. 334–337.

(Kovalev et al 2016) Vassili Kovalev, Alexander Kalinovsky, Sergey Kovalev, 2016. Deep Learning with Theano, Torch, Caffe, TensorFlow, and Deeplearning4J: Which One Is the Best in Speed and Accuracy?

(Pycharm 2017) Pycharm, JetBrains, visto a 05 Fevereiro 2017, https://www.jetbrains.com/pycharm/

(Piotr et al 2008) Piotr W. Mirowski, Member, IEEE, Yann LeCun, Deepak Madhavan, and Ruben Kuzniecky, 2008. Comparing SVM and Convolutional Networks for Epileptic Seizure Prediction from Intracranial EEG

(Data Mining 2017) Data Mining, Oracle, visto a 07 Fevereiro

2017,https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/process.htm#DMCON002

(Al-Rfou et al 2016) Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merrienboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schluter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang,2016. Theano: A Python framework for fast computation of mathematical expressions

(Woodall 2003) Tony Woodall, 2003. Conceptualising 'Value for the Customer' : An Attributional, Structural and Dispositional Analysis

(Lapierre 2000) Jozée Lapierre, 2000. Customer-perceived value in industrial contexts

(Nicola et al 2012) Susana Nicola, Eduarda Pinto Ferreira, J. J. Pinto Ferreira, 2012. International Journal of Information Technology & Decision Making 11:03, 661-703.

Dornhege et al 2007 Guido Dornhege, José del R. Millán, Thilo Hinterberger, Dennis J. McFarland, Klaus-Robert Muller, 2007. Toward Brain-Computer Interfacing, A Bradford Book.

(Al-Fahoum et al 2014) Amjed S. Al-Fahoum, Ausilah A. Al-Fraihat, 2014. Methods of EEG Signal Features Extraction Using Linear Analysis in Frequency and Time-Frequency Domains ISRN Neuroscience, vol. 2014, Article ID 730218, 7 pages. doi:10.1155/2014/730218

(Lotte et al 2007) Fabien Lotte, Marco Congedo, Anatole Lécuyer, Fabrice Lamarche, Bruno Arnaldi, 2007. A review of classification algorithms for EEG-based brain-computer interfaces Journal of Neural Engineering, IOP Publishing, 4, pp.24.<inria-00134950>

(Bandarabadi et al 2015) Mojtaba Bandarabadi, César A. Teixeira, Jalil Rasekhi, António Dourado, 2015. Epileptic seizure prediction using relative spectral power features.

(Mirowski et al 2009) Piotr Mirowski, Deepak Madhavan, Yann LeCun, Ruben Kuzniecky, 2009. Classification of patterns of EEG synchronizations for seizure prediction

(Kevric et al 2012) Jasmin Kevric, Abdulhamit Subasi, 2012. Classification of EEG signals for epileptic seizure prediction using ANN.

(Mirowski et al 2008) Piotr W. Mirowski, *Member*, *IEEE*, Yann LeCun, Deepak Madhavan, Ruben Kuzniecky, 2008. Comparing SVM and Convolutional Networks for

Epileptic Seizure Prediction from Intracranial EEG.

(Park et al 2011) Yun Park, Lan Luo, Keshab K. Parhi, Theoden Netoff, 2011. Seizure prediction with spectral power of EEG using cost-sensitive support vector machines

(Schindler et al 2007) Kaspar Schindler, Howan Leung, Christian E. Elger, Klaus Lehnertz, 2007. Assessing seizure dynamics by analysing the correlation structure of multichannel intracranial EEG.

(Antoniades et al 2016) Andreas Antoniades, Loukianos Spyrou, Clive Cheong Took, Saeid Sanei, 2016. Deep Learning For Epileptic Intracranial EEG Data.

(Page et al 2014) Adam Page, JT Turner, Tinoosh Mohsenin and Tim Oates, 2014. Comparing Raw Data and Feature Extraction for Seizure Detection with Deep Learning Methods.

(Kaggle Am 2016) Kaggle, American Epilepsy Society Seizure Prediction Challenge, visto a 10 Dezembro 2016, https://www.kaggle.com/c/seizure-prediction.

(EpilepsyColorado 2017) EpilepsyColorado, Epilepsy Foundation Types of Seizures, visto a 24 Fevereiro 2017, http://www.epilepsycolorado.org/wp-content/uploads/2016/01/2-Types-of-Seizures.pdf

(Stanford CNN 2017) Stanford, Convolutional Neural Network, visto a 25 Fevereiro 2017, http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

(Stanford NN 2017) Stanford, Multi-Layer Neural Network, visto a 25 Fevereiro 2017, http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

(Stanford LR 2017) Stanford, Logistic Regression, visto a 25 Fevereiro 2017, http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/

(DataScienceCentral 2017) DataScienceCentral, Random Forests Algorithm, visto a 25 Fevereiro 2017, http://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm

(Statistica 2017) Statistica, Random Forests, visto a 25 Fevereiro 2017, http://www.statsoft.com/Textbook/Random-Forest

(Demsar 2006) Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research*, 7, pp.1-30.

(Gama et al 2012) Gama, J., Carvalho, A., Oliveira, M., Faceli, K., Lorena, A., 2012. Extração de Conhecimento de Dados, Data Mining. JC Gama Extração Conhecimento Dados Data Mining.

(Casson et al) Alexander J. Casson, Elena Luna, Esther Rodriguez-Villegas, Performance metrics for the accurate characterisations of interictal spike detection algorithms.

(Nvidia 2017) Nvidia, Deep Learning, visto a 25 Fevereiro 2017, https://developer.nvidia.com/deep-learning

(Nurse et al 2016) Ewan Nurse, Benjamin S. Mashford, Antonio Jimeno Yepes, Isabell Kiral-Kornek, Stefan Harrer, Dean R. Freestone, 2016. Decoding EEG and LFP Signals using Deep Learning: Heading TrueNorth

(LeCun et al 1998) Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, pp. 2278–2324, Nov 1998. pages 8, 9, 15

(Walker 2015) Ian Walker, 2015. Deep Convolutional Neural Networks for Brain Computer Interface using Motor Imagery

(Tabar et al 2016) Yousef Rezaei Tabar, Ugur Halici, 2016. A novel deep learning approach for classification of EEG motor imagery signals

(Audacity 2016) Audacity, visto a 03 Março 2017, www.audacityteam.org/

(Dauwels et al) Justin Dauwels, François Vialatte, Andrzej Cichocki, Diagnosis of Alzheimer's Disease from EEG Signals: Where Are We Standing

(CS231n CNN 2017) CS231n. CS231n Convolutional Neural Networks for Visual Recognition, visto a

04 Março 2017, http://cs231n.github.io/convolutional-networks/

(Rumelhart et al 1988) Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3), 1

(Goodfellow et al 2016) Ian Goodfellow, Yoshua Bengio, Aaron Courville, 2016. Deep Learning, MIT Press http://www.deeplearningbook.org

(Koen et al) Peter A. Koen, Greg M. Ajamian, Scott Boyce, Allen Clamen, Eden Fisher, Stavros Fountoulakis, Albert Johnson, Pushpinder Puri, Rebecca Seibert. Fuzzy Front End: Effective Methods, Tools and Techniques.

(CS231n NN 2017) (CS231n. CS231n Convolutional Neural Networks for Visual Recognition, visto a 04 Maio 2017, http://cs231n.github.io/neural-networks-1/

(Martins 2016) Martins, Helder, 2016. Deteção de convulsões epiléticas a partir de eletroencefalogramas

(Peng et al 2002) Peng, C.-Y.J., Lee, K.L., Ingersoll, G.M., 2002. An Introduction to Logistic Regression Analysis and Reporting. J. Educ. Res. 96, 3–14. doi:10.1080/00220670209598786

(MachineLearningMistery ADM 2017) MachineLearningMistery, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, visto a 19 Outubro 2017, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

(MachineLearningMistery GRD DSC 2017) MachineLearningMistery, Gradient Descent For Machine Learning, visto a 19 Outubro 2017, https://machinelearningmastery.com/gradient-descent-for-machine-learning/

8 Anexos

8.1 Anexo 1

A Tabela 16, contém os resultados obtidos para o *dataset* Paciente 1, com as imagens separadas por canais.

Tabela 16 - Resultados Paciente 1 imagens separadas por canais

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
1	Relu	Adam optimizer	0.769	0.59	0.98/0.2	0.775/0.25	0.861/0.216
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.780	0.615	0.98/0.25	0.795/0.25	0.870/0.233
	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.921/0.9	0.956/0.8
	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.623	0.51	0.72/0.3	0.623/0.090	0.631/0.138
2	Relu	Adam optimizer	0.780	0.655	0.96/0.35	0.811/0.333	0.865/0.316
	Sigmoid	Adam optimizer	0.709	0.495	0.94/0.05	0.733/0.025	0.815/0.033
	Softplus	Adam optimizer	0.766	0.61	0.92/0.3	0.823/0.206	0.844/0.237
	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.957/0.766
	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
3	Relu	Adam optimizer	0.680	0.585	0.82/0.35	0.703/0.161	0.744/0.217
	Sigmoid	Adam	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044

Canal	Função ativação	Função treino optimizer	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
4	Softplus	Adam optimizer	0.738	0.605	0.86/0.35	0.735/0.278	0.781/0.277
	Tanh	Adam optimizer	0.840	0.74	0.98/0.5	0.852/0.55	0.904/0.5
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.582	0.46	0.62/0.3	0.489/0.073	0.544/0.117
	Relu	Adam optimizer	0.711	0.635	0.82/0.45	0.832/0.266	0.780/0.323
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.726	0.605	0.86/0.35	0.820/0.233	0.795/0.266
	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.957/0.766
5	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133
	Relu	Adam optimizer	0.669	0.57	0.84/0.3	0.8/0.111	0.762/0.153
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.785	0.67	0.94/0.4	0.820/0.4	0.862/0.383
	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.909/0.8	0.948/0.733
6	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Relu	Adam optimizer	0.690	0.565	0.88/0.25	0.775/0.1	0.8/0.14

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
7	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.709	0.565	0.88/0.25	0.791/0.133	0.793/0.166
	Tanh	Adam optimizer	0.926	0.89	0.98/0.8	0.938/0.85	0.954/0.8
	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Relu	Adam optimizer	0.638	0.475	0.8/0.15	0.640/0.048	0.699/0.073
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.723	0.55	0.9/0.2	0.753/0.17	0.807/0.178
	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.909/0.8	0.948/0.733
	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Relu	Adam optimizer	0.635	0.58	0.76/0.4	0.728/0.176	0.690/0.219
8	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.797	0.685	0.92/0.45	0.852/0.406	0.861/0.403
	Tanh	Adam optimizer	0.885	0.815	0.98/0.65	0.897/0.666	0.929/0.646
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133
9	Relu	Adam	0.623	0.54	0.78/0.3	0.8/0.08	0.707/0.123

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
		optimizer					
	Sigmoid	Adam optimizer	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Softplus	Adam optimizer	0.723	0.565	0.88/0.25	0.807/0.183	0.797/0.183
	Tanh	Adam optimizer	0.869	0.765	0.98/0.55	0.880/0.566	0.920/0.546
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133
	Relu	Adam optimizer	0.638	0.55	0.8/0.3	0.7/0.091	0.715/0.135
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.702	0.535	0.92/0.15	0.763/0.125	0.811/0.106
	Tanh	Adam optimizer	0.928	0.875	1.0/0.75	0.926/0.8	0.957/0.766
10	Relu	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Sigmoid	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Relu	Adam optimizer	0.769	0.605	0.96/0.25	0.786/0.233	0.857/0.24
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.738	0.575	0.9/0.25	0.803/0.183	0.811/0.2
	Tanh	Adam optimizer	0.885	0.8	1.0/0.6	0.885/0.6	0.933/0.6
11	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
12	Relu	Adam optimizer	0.723	0.605	0.86/0.35	0.807/0.215	0.803/0.257
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.738	0.56	0.92/0.2	0.773/0.191	0.828/0.18
	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.905/0.8	0.948/0.733
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.542	0.5	0.6/0.4	0.440/0.102	0.507/0.161
13	Relu	Adam optimizer	0.769	0.62	0.94/0.3	0.802/0.35	0.853/0.3
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Adam optimizer	0.885	0.815	0.98/0.65	0.892/0.7	0.930/0.666
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.652	0.5	0.8/0.2	0.595/0.057	0.681/0.088
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
14	Relu	Adam optimizer	0.754	0.625	0.9/0.35	0.812/0.34	0.832/0.323
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.709	0.57	0.84/0.3	0.819/0.266	0.763/0.233
	Tanh	Adam optimizer	0.914	0.865	0.98/0.75	0.918/0.85	0.945/0.783
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117
	Softplus	Gradient descent	0.695	0.5	0.9/0.1	0.666/0.028	0.765/0.044
	Tanh	Gradient	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133

Canal	Função ativação	Função treino descent	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
15	Relu	Adam optimizer	0.695	0.56	0.82/0.3	0.723/0.161	0.731/0.194
	Sigmoid	Adam optimizer	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Softplus	Adam optimizer	0.766	0.595	0.94/0.25	0.807/0.216	0.853/0.213
	Tanh	Adam optimizer	0.914	0.85	1.0/0.7	0.914/0.7	0.95/0.7
	Relu	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Sigmoid	Gradient descent	0.609	0.5	0.7/0.3	0.523/0.085	0.598/0.133
	Softplus	Gradient descent	0.738	0.5	1.0/0.0	0.738/0.0	0.848/0.0
	Tanh	Gradient descent	0.585	0.5	0.7/0.3	0.511/0.073	0.590/0.117

8.2 Anexo 2

A Tabela 17, contém os resultados obtidos para o *dataset* Paciente 2, com as imagens separadas por canais.

Tabela 17 - Resultados Paciente 2 imagens separadas por canais

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
1	Relu	Adam optimizer	0.681	0.575	0.8/0.35	0.716/0.24	0.719/0.253
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.721	0.632	0.815/0.45	0.839/0.273	0.791/0.333
	Tanh	Adam optimizer	0.926	0.887	0.975/0.8	0.942/0.766	0.952/0.78
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.717	0.525	1.0/0.05	0.714/0.1	0.832/0.066
	Tanh	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
2	Relu	Adam optimizer	0.696	0.637	0.825/0.45	0.718/0.328	0.744/0.344
	Sigmoid	Adam	0.645	0.49	0.88/0.1	0.626/0.028	0.730/0.044

Canal	Função ativação	Função treino optimizer	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
3	Softplus	Adam optimizer	0.686	0.537	0.875/0.2	0.749/0.173	0.770/0.163
	Tanh	Adam optimizer	0.909	0.865	0.98/0.75	0.91/0.9	0.941/0.8
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.719	0.575	0.9/0.25	0.777/0.166	0.811/0.196
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.640	0.525	0.85/0.2	0.646/0.078	0.724/0.111
	Tanh	Adam optimizer	0.822	0.737	0.975/0.5	0.836/0.55	0.890/0.5
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Relu	Adam optimizer	0.778	0.667	0.935/0.4	0.818/0.316	0.852/0.346
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.752	0.587	0.975/0.2	0.751/0.35	0.846/0.25
4	Tanh	Adam optimizer	0.942	0.9	1.0/0.8	0.942/0.8	0.966/0.8
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
5	Relu	Adam optimizer	0.752	0.625	0.9/0.35	0.816/0.3	0.826/0.3

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
6	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.819	0.712	0.975/0.45	0.836/0.45	0.890/0.433
	Tanh	Adam optimizer	0.892	0.837	0.975/0.7	0.902/0.766	0.930/0.713
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.519	0.49	0.58/0.4	0.413/0.115	0.481/0.177
	Relu	Adam optimizer	0.738	0.577	0.955/0.2	0.753/0.25	0.837/0.216
	Sigmoid	Adam optimizer	0.736	0.55	1.0/0.1	0.736/0.1	0.844/0.1
	Softplus	Adam optimizer	0.719	0.575	0.9/0.25	0.769/0.183	0.808/0.206
	Tanh	Adam optimizer	0.926	0.875	1.0/0.75	0.922/0.8	0.955/0.766
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Relu	Adam optimizer	0.569	0.537	0.675/0.4	0.609/0.128	0.593/0.19
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.769	0.687	0.875/0.5	0.856/0.333	0.839/0.393
7	Tanh	Adam optimizer	0.942	0.9	1.0/0.8	0.942/0.8	0.966/0.8
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.717	0.54	0.98/0.1	0.731/0.066	0.83/0.08
	Tanh	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
8	Relu	Adam	0.769	0.65	0.9/0.4	0.829/0.4	0.835/0.366

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
		optimizer					
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.736	0.6	0.9/0.3	0.802/0.183	0.822/0.226
	Tanh	Adam optimizer	0.876	0.8	1.0/0.6	0.876/0.6	0.926/0.6
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.669	0.5	0.9/0.1	0.722/0.05	0.784/0.066
	Sigmoid	Adam optimizer	0.702	0.525	0.95/0.1	0.736/0.05	0.811/0.066
	Softplus	Adam optimizer	0.606	0.537	0.775/0.3	0.776/0.09	0.686/0.137
	Tanh	Adam optimizer	0.926	0.875	1.0/0.75	0.922/0.8	0.955/0.766
9	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.516	0.487	0.575/0.4	0.411/0.115	0.478/0.177
	Relu	Adam optimizer	0.724	0.567	0.935/0.2	0.744/0.3	0.825/0.233
	Sigmoid	Adam optimizer	0.636	0.5	0.8/0.2	0.569/0.066	0.664/0.1
	Softplus	Adam optimizer	0.566	0.525	0.7/0.35	0.489/0.173	0.575/0.183
	Tanh	Adam optimizer	0.911	0.892	0.935/0.85	0.946/0.8	0.935/0.82
10	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
11	Relu	Adam optimizer	0.719	0.562	0.925/0.2	0.752/0.166	0.820/0.18
	Sigmoid	Adam optimizer	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Softplus	Adam optimizer	0.719	0.575	0.9/0.25	0.741/0.275	0.801/0.25
	Tanh	Adam optimizer	0.909	0.862	0.975/0.75	0.922/0.766	0.941/0.746
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.560	0.5	0.6/0.4	0.431/0.128	0.501/0.194
12	Relu	Adam optimizer	0.626	0.525	0.8/0.25	0.656/0.103	0.699/0.14
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.696	0.575	0.9/0.25	0.664/0.228	0.761/0.211
	Tanh	Adam optimizer	0.897	0.885	0.92/0.85	0.93/0.866	0.922/0.846
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.490	0.5	0.5/0.5	0.346/0.143	0.408/0.222
13	Relu	Adam optimizer	0.686	0.562	0.825/0.3	0.702/0.15	0.736/0.196
	Sigmoid	Adam optimizer	0.669	0.5	0.9/0.1	0.636/0.033	0.744/0.05
	Softplus	Adam optimizer	0.643	0.547	0.845/0.25	0.714/0.153	0.745/0.178
	Tanh	Adam optimizer	0.895	0.84	0.98/0.7	0.898/0.8	0.933/0.733
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient	0.547	0.51	0.62/0.4	0.518/0.12	0.525/0.183

Canal	Função ativação	Função treino descent	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
14	Relu	Adam optimizer	0.702	0.587	0.825/0.35	0.716/0.25	0.745/0.263
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.606	0.525	0.75/0.3	0.584/0.12	0.647/0.163
	Tanh	Adam optimizer	0.940	0.912	0.975/0.85	0.954/0.866	0.959/0.846
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
15	Relu	Adam optimizer	0.669	0.6	0.75/0.45	0.636/0.283	0.675/0.313
	Sigmoid	Adam optimizer	0.719	0.562	0.925/0.2	0.769/0.116	0.816/0.146
	Softplus	Adam optimizer	0.642	0.6	0.75/0.45	0.738/0.281	0.683/0.322
	Tanh	Adam optimizer	0.909	0.85	1.0/0.7	0.909/0.7	0.946/0.7
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
16	Relu	Adam optimizer	0.710	0.64	0.83/0.45	0.81/0.366	0.778/0.373
	Sigmoid	Adam optimizer	0.688	0.49	0.98/0.0	0.698/0.0	0.813/0.0
	Softplus	Adam optimizer	0.652	0.537	0.775/0.3	0.686/0.173	0.694/0.207
	Tanh	Adam optimizer	0.883	0.845	0.94/0.75	0.906/0.8	0.921/0.766
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
17	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.732	0.662	0.825/0.5	0.764/0.433	0.773/0.438
	Sigmoid	Adam optimizer	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Softplus	Adam optimizer	0.707	0.66	0.82/0.5	0.74/0.55	0.74/0.475
	Tanh	Adam optimizer	0.845	0.805	0.91/0.7	0.893/0.666	0.890/0.666
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
18	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.756	0.675	0.85/0.5	0.849/0.456	0.805/0.437
	Sigmoid	Adam optimizer	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Softplus	Adam optimizer	0.722	0.65	0.85/0.45	0.817/0.3	0.800/0.35
	Tanh	Adam optimizer	0.907	0.877	0.955/0.8	0.926/0.866	0.936/0.813
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
19	Tanh	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Relu	Adam optimizer	0.639	0.562	0.825/0.3	0.676/0.12	0.719/0.163
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.659	0.575	0.85/0.3	0.676/0.136	0.738/0.18
	Tanh	Adam optimizer	0.875	0.802	0.955/0.65	0.898/0.666	0.919/0.646
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
		descent					
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.736	0.55	1.0/0.1	0.729/0.2	0.842/0.133
	Sigmoid	Adam optimizer	0.702	0.55	0.9/0.2	0.769/0.106	0.790/0.137
	Softplus	Adam optimizer	0.686	0.562	0.825/0.3	0.702/0.15	0.736/0.196
	Tanh	Adam optimizer	0.942	0.9	1.0/0.8	0.942/0.8	0.966/0.8
20	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.593	0.5	0.7/0.3	0.498/0.095	0.581/0.144
	Relu	Adam optimizer	0.826	0.762	0.925/0.6	0.864/0.6	0.877/0.583
	Sigmoid	Adam optimizer	0.652	0.512	0.825/0.2	0.669/0.073	0.704/0.107
	Softplus	Adam optimizer	0.669	0.537	0.825/0.25	0.682/0.173	0.713/0.173
	Tanh	Adam optimizer	0.895	0.865	0.93/0.8	0.938/0.733	0.927/0.76
21	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.686	0.562	0.825/0.3	0.802/0.14	0.757/0.190
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.719	0.6	0.85/0.35	0.809/0.29	0.786/0.290
22	Tanh	Adam optimizer	0.907	0.875	0.95/0.8	0.934/0.833	0.934/0.793
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133

Canal	Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
23	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.632	0.562	0.725/0.4	0.822/0.153	0.682/0.221
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.736	0.625	0.85/0.4	0.829/0.356	0.794/0.337
	Tanh	Adam optimizer	0.876	0.812	0.975/0.65	0.889/0.666	0.921/0.646
	Relu	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Sigmoid	Gradient descent	0.576	0.5	0.7/0.3	0.489/0.086	0.575/0.133
	Softplus	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Tanh	Gradient descent	0.533	0.5	0.6/0.4	0.418/0.115	0.492/0.177
	Relu	Adam optimizer	0.769	0.65	0.9/0.4	0.824/0.366	0.840/0.363
	Sigmoid	Adam optimizer	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Softplus	Adam optimizer	0.736	0.662	0.875/0.45	0.817/0.333	0.815/0.366
24	Tanh	Adam optimizer	0.859	0.8	0.95/0.65	0.889/0.65	0.902/0.633
	Relu	Gradient descent	0.660	0.5	0.9/0.1	0.631/0.028	0.741/0.044
	Sigmoid	Gradient descent	0.566	0.5	0.7/0.3	0.484/0.081	0.572/0.127
	Softplus	Gradient descent	0.702	0.5	1.0/0.0	0.702/0.0	0.824/0.0
	Tanh	Gradient descent	0.473	0.5	0.45/0.55	0.346/0.177	0.386/0.262

8.3 Anexo 3

No *dataset* Cão 1, existem 480 exemplos da classe *interictal* e 24 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.952.

Tabela 18 - Resultados Cão 1 para as imagens completas com *shuffle*

Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
Relu	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Sigmoid	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Softplus	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Tanh	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Relu	Gradient descent	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Sigmoid	Gradient descent	0.860	0.5	0.9/0.1	0.856/0.004	0.877/0.007
Softplus	Gradient descent	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Tanh	Gradient descent	0.772	0.5	0.8/0.2	0.762/0.009	0.780/0.018

No *dataset* Cão 2, existem 500 exemplos da classe *interictal* e 42 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.922.

Tabela 19 - Resultados Cão 2 para as imagens completas com *shuffle*

Função ativação	Função treino	*Accuracy*	AUC	*Recall*	*Precision*	*F-Measure*
Relu	Adam optimizer	0.922	0.5	1.0/0.0	0.922/0.0	0.959/0.0
Sigmoid	Adam optimizer	0.922	0.5	1.0/0.0	0.922/0.0	0.959/0.0
Softplus	Adam optimizer	0.922	0.5	1.0/0.0	0.922/0.0	0.959/0.0
Tanh	Adam optimizer	0.929	0.584	0.994/0.175	0.934/0.325	0.963/0.217
Relu	Gradient descent	0.840	0.5	0.9/0.1	0.831/0.009	0.864/0.016
Sigmoid	Gradient descent	0.837	0.5	0.9/0.1	0.829/0.007	0.863/0.013
Softplus	Gradient descent	0.840	0.5	0.9/0.1	0.831/0.009	0.864/0.016
Tanh	Gradient descent	0.755	0.5	0.8/0.2	0.739/0.016	0.768/0.030

No *dataset* Cão 3, existem 1440 exemplos da classe *interictal* e 72 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.952.

Tabela 20 - Resultados Cão 3 para as imagens completas com *shuffle*

Função ativação	**Função treino**	***Accuracy***	**AUC**	***Recall***	***Precision***	***F-Measure***
Relu	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Sigmoid	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Softplus	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Tanh	Adam optimizer	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Relu	Gradient descent	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Sigmoid	Gradient descent	0.861	0.5	0.9/0.1	0.857/0.004	0.877/0.008
Softplus	Gradient descent	0.952	0.5	1.0/0.0	0.952/0.0	0.975/0.0
Tanh	Gradient descent	0.861	0.5	0.9/0.1	0.857/0.004	0.877/0.008

No *dataset* Cão 4, existem 804 exemplos da classe *interictal* e 97 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.892.

Tabela 21 - Resultados Cão 4 para as imagens completas com *shuffle*

Função ativação	**Função treino**	***Accuracy***	**AUC**	***Recall***	***Precision***	***F-Measure***
Relu	Adam optimizer	0.892	0.5	1.0/0.0	0.892/0.0	0.943/0.0
Sigmoid	Adam optimizer	0.892	0.5	1.0/0.0	0.892/0.0	0.943/0.0
Softplus	Adam optimizer	0.892	0.5	1.0/0.0	0.892/0.0	0.943/0.0
Tanh	Adam optimizer	0.883	0.509	0.986/0.033	0.894/0.042	0.937/0.037
Relu	Gradient descent	0.814	0.5	0.9/0.1	0.803/0.010	0.848/0.019
Sigmoid	Gradient descent	0.814	0.5	0.9/0.1	0.803/0.010	0.848/0.019
Softplus	Gradient descent	0.814	0.5	0.9/0.1	0.803/0.010	0.848/0.019
Tanh	Gradient descent	0.658	0.5	0.7/0.3	0.625/0.033	0.660/0.059

No *dataset* Cão 5, existem 450 exemplos da classe *interictal* e 30 da classe *preictal* (conforme a Tabela 4). Assim, para que o resultado seja interessante, a *Accuracy* deverá ser superior a 0.937.

Tabela 22 - Resultados Cão 5 para as imagens completas com *shuffle*

Função ativação	**Função treino**	***Accuracy***	**AUC**	***Recall***	***Precision***	***F-Measure***
Relu	Adam optimizer	0.937	0.5	1.0/0.0	0.937/0.0	0.967/0.0
Sigmoid	Adam optimizer	0.937	0.5	1.0/0.0	0.937/0.0	0.967/0.0
Softplus	Adam optimizer	0.937	0.5	1.0/0.0	0.937/0.0	0.967/0.0
Tanh	Adam optimizer	0.941	0.548	0.997/0.1	0.943/0.1	0.969/0.1
Relu	Gradient descent	0.85	0.5	0.9/0.1	0.843/0.006	0.870/0.011
Sigmoid	Gradient descent	0.85	0.5	0.9/0.1	0.843/0.006	0.870/0.011
Softplus	Gradient descent	0.85	0.5	0.9/0.1	0.843/0.006	0.870/0.011
Tanh	Gradient descent	0.762	0.5	0.8/0.2	0.75/0.012	0.774/0.023